DECODES

Device Conversion and Delivery System

Version 5.4

User's Guide

prepared for

U.S. Geological Survey, Water Resources Division

and

U.S. Army Corps of Engineers

prepared by

Ilex Engineering, Inc
5114 Crystal Park Lane
Ellicott City, MD 21043
Tel: 410.465.6948
Email: info@ilexeng.com

Revision 1.0

July 28, 2003

Table of Contents

1. INTRODUCTION TO DECODES 1

2. DECODES GENERAL CONCEPTS 3

3. DECODES DATABASE ARCHITECTURE 5

3.1 SETUP INFORMATION 5
3.1.1 Enumerations 5
3.1.2 Time Zones 7
3.1.3 Sensor Data Types 7
3.1.4 Engineering Units 8
3.2 PROPERTIES 8
3.3 DECODING, FORMATTING, AND CONVERTING SPECIFICATIONS 9
3.3.1 Equipment Models 9
3.4 SITES & SITE NAMES 10
3.5 PLATFORM INFORMATION 11
3.6 PERFORMANCE MEASUREMENTS 12
3.7 PRESENTATION AND UNIT CONVERSION 13
3.8 EQUATION PROCESSOR INFORMATION 15
3.9 ROUTING SPECIFICATION INFORMATION 17

4. INSTALLING OR UPGRADING DECODES 18

4.1 INSTALLING JAVA 18
4.1.1 Installing Java on Windows 18
4.1.2 Installing Java under Red Hat Linux 19
4.2 INSTALLING THE COMPLETE DECODES PACKAGE 20
4.2.1 Setup DECODES Runtime Environment 21
4.2.2 Setup your DECODES Properties 22
4.3 UPGRADE INSTALLATION 23

5. USING DECODES WITH AN SQL DATABASE 24

5.1 POSTGRESQL 24
5.1.1 Installing PostgreSQL 24
5.1.1.1 Installing PostgreSQL on Windows 2000, NT, or XP 25
5.1.2 PostgreSQL Startup and Initialization 25
5.1.2.1 Start the PostgreSQL Server on Windows 2000 or NT 26
5.1.3 Create PostgreSQL Administrative Account 26
5.1.4 Setup PostgreSQL Security 26
5.2 HOW DECODES DOES DATABASE AUTHENTICATION 27
5.3 CONFIGURE DECODES FOR AN SQL EDITABLE DATABASE 28
5.4 CREATING THE DECODES EDITABLE DATABASE 28
5.5 IMPORTING DATA FROM YOUR OLD XML DATABASE 30
5.6 VERIFY THAT THE DATABASE WORKS 31

6. MAINTAINING THE DECODES DATABASE 32

6.1 INITIALIZING YOUR EDITABLE DECODES DATABASE 32
6.1.1 Standard XML Setup Files 32
6.1.2 Importing Data from EMIT or Pre-Release-5 DECODES 33
6.1.3 Importing XML Data from Other DECODES Sites 34
6.2 INTERACTIVELY EDITING THE DATABASE 34
6.3 UPDATING THE INSTALLED DATABASE 35
6.4 EXPORTING PLATFORMS TO XML FILES 36
6.5 OTHER DATABASE UTILITIES 37
6.5.1 Creating the Platform Cross-Reference File 37
6.5.2 Creating LRGS-Style Network List Files 38

7. THE DECODES DATABASE EDITOR 39

7.1 GUI ORGANIZATION 40
7.1.1 List Panels in General 40
7.1.2 Edit Panels in General 40
7.1.3 Exiting the Editor 40
7.2 THE PLATFORM EDIT PANEL 42
7.3 THE SITE EDIT PANEL 44
7.4 THE PLATFORM-CONFIG EDIT PANEL 45
7.4.1 The Decoding Script Edit Dialog 48
7.5 THE EQUIPMENT-MODEL EDIT PANEL 50
7.6 THE PRESENTATION GROUP EDIT PANEL 51
7.7 THE DATA SOURCE EDIT PANEL 53
7.8 THE NETWORK LIST EDIT PANEL 55
7.9 THE ROUTING SPECIFICATION EDIT PANEL 56

8. THE DECODES FORMAT LANGUAGE 58

8.1 EXECUTION OF FORMAT STATEMENTS BY A ROUTING SPEC 58
8.2 STEPPING THROUGH THE SCRIPT AND THE DATA 58
8.3 FORMAT OPERATIONS 59
8.3.1 Skipping and Positioning Operations 61
8.3.2 The Check Operation 61
8.3.3 The Scan Operation 62
8.3.4 The Jump Operation 62
8.3.5 The Repeat Operation 62
8.3.6 Field Operations 63
8.3.6.1 Date Fields 64
8.3.6.2 Time Fields 65
8.3.6.3 Time Interval Fields 66
8.3.6.4 Format Label Fields 66
8.3.6.5 Sensor Value Fields 67

9. DECODES ROUTING SPECIFICATIONS 68

9.1 HOW TO RUN A ROUTING SPECIFICATION 69
9.1.1 Routing Spec Properties 69
9.2 DATA SOURCES 70
9.2.1 LRGS Data Source 70
9.2.1.1 Timeouts in LRGS Data Sources 72
9.2.2 File Data Source 73
9.2.2.1 Delimiting Messages Within the File 73
9.2.3 Directory Data Source 73
9.2.4 Hot Backup Group Data Source 74
9.2.5 Round Robin Group Data Source 75
9.2.6 Socket Stream Data Source 76
9.2.6.1 Using SocketStreamDataSource for NOAAPORT 77
9.3 OUTPUT FORMATTERS 78
9.3.1 SHEF Output Format 79
9.3.2 SHEFIT Output Format 80
9.3.3 Human Readable Output Format 81
9.3.4 EMIT-ASCII Format 82
9.3.5 EMIT-Oracle Format 83
9.3.6 Dump Formatter 84
9.3.7 USGS STDFMT Output Formatter 85
9.3.8 Transmit Monitor Formatter 86
9.4 CONSUMERS 88
9.4.1 Pipe Consumer 88
9.4.2 File Consumer 88

9.4.3 Directory Consumer *89*
9.5 TIME TAGGING DATA SAMPLES 91

10. SPECIFIC SCENARIOS 92

10.1 HOW TO CREATE A NEW PLATFORM SPECIFICATION 92
Create a Site for the New Platform *92*
Create an Equipment Model Record for the New Platform *92*
Create a Configuration for the New Platform *93*
Rules and Conventions for Configuration Naming 94
Enter the Sensor and Formatting Information 95
Create the Platform Record *96*
Add the new Platform to a Network List *97*
Testing the new Platform in a Routing Spec *97*

Table of Figures

FIGURE 1-1: WHAT DECODES DOES. 1
FIGURE 2-1: DECODES DETAILED DATA FLOW 3
FIGURE 2-2: DECODES COMPONENTS. 4
FIGURE 3-1: ENUMERATIONS ERD. 5
FIGURE 3-2: DATA TYPES ERD. 7
FIGURE 3-3: ENGINEERING UNITS & CONVERTERS ERD. 8
FIGURE 3-4: EQUIPMENT MODEL ERD. 9
FIGURE 3-5: SITES AND SITE NAMES ERD. 10
FIGURE 3-6: PLATFORM INFORMATION ERD. 11
FIGURE 3-7: PERFORMANCE MEASUREMENTS ERD. 12
FIGURE 3-8: PRESENTATION AND UNIT CONVERSION ERD. 13
FIGURE 3-9: EQUATION PROCESSOR ERD. 15
FIGURE 3-10: LOOKUP TABLES ERD. 16
FIGURE 3-11: ROUTING SPECIFICATION ERD. 17
FIGURE 4-1: WINDOWS 2000 ENVIRONMENT VARIABLE DIALOG. 21
FIGURE 7-1: DATABASE EDITOR PLATFORM LIST SCREEN. 39
FIGURE 7-2: PLATFORM CONFIG EDIT PANEL. 41
FIGURE 7-3: PLATFORM EDIT PANEL. 43
FIGURE 7-4: TRANSPORT MEDIUM EDIT DIALOG. 43
FIGURE 7-5: THE SITE EDIT PANEL. 44
FIGURE 7-6: PLATFORM CONFIG EDIT PANEL. 46
FIGURE 7-7: EDIT CONFIG SENSOR DIALOG. 47
FIGURE 7-8: DECODING SCRIPT EDIT DIALOG SHOWING INTERACTIVE DECODING. 49
FIGURE 7-9: EQUIPMENT MODEL EDIT DIALOG. 50
FIGURE 7-10: PRESENTATION GROUP EDIT PANEL. 52
FIGURE 7-11: DATA SOURCE EDIT PANEL SHOWING LRGS DATA SOURCE. 53
FIGURE 7-12: DATA SOURCE EDIT PANEL SHOWING HOT BACKUP GROUP. 54
FIGURE 7-13: NETWORK LIST EDIT PANEL. 55
FIGURE 7-14: ROUTING SPECIFICATION EDIT PANEL. 57
FIGURE 9-1: DATA FLOW FOR ROUTING SPECIFICATIONS. 68
FIGURE 9-2: EXAMPLE OF SHEF .A OUTPUT 79
FIGURE 9-3: EXAMPLE OF SHEF .E OUTPUT 79
FIGURE 9-4: EXAMPLE OF SHEFIT OUTPUT FORMAT. 80
FIGURE 9-5: EXAMPLE OF HUMAN READABLE OUTPUT FORMAT. 81
FIGURE 9-6: EXAMPLE OF EMIT-ASCII FORMAT. 82
FIGURE 9-7: EXAMPLE OF EMIT-ORACLE OUTPUT FORMAT. 83
FIGURE 9-8: EXAMPLE OF DUMP OUTPUT FORMAT 84
FIGURE 9-9: EXAMPLE OF USGS STDFMT OUTPUT. 85
FIGURE 9-10: EXAMPLE OF TRANSMIT MONITOR FORMAT. 86

1. Introduction to DECODES

DECODES stands for DEvice COnversion and DElivery System. DECODES is a suite of software that takes data from a variety of recording devices and converts it into standard engineering units, suitable for entry into a database.

The types of recording devices include both Electronic Data Loggers (EDLs), which are electronic recorders whose data for the most part are manually retrieved, and Data Collection Platforms (DCPs), whose data are retrieved by satellite telemetry.

The operations performed by DECODES are depicted in Figure 1-1.

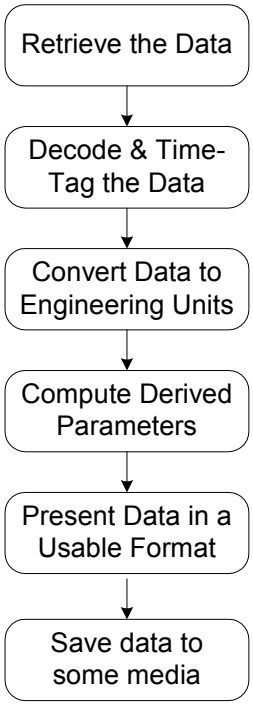

Figure 1-1: What DECODES Does.

The fourth step, “Compute Derived Parameters” is not implemented in the current release.

Currently DECODES can handle data from any GOES DCP received either from a GOES receiver or over DOMSAT. It has the basic capability to parse data from EDL files, but this capability has not been extensively tested in the current release.

Data can be retrieved from saved files or over the network from an LRGS.

DECODES can handle any ASCII format currently in use by the DCS. This includes true ASCII values or the pseudo-binary values common in compact random messages.

DECODES uses a database of platform specifications to tell it how to decode data from a given source. DECODES manages a fairly complex database that includes entities for:

DataSource	Where to retrieve raw data from: Directory, File, LRGS Network Connection, shared memory, etc. If an LRGS network connection is specified, you can specify network lists, time-ranges, etc.
Platforms & Sites	Site-specific parameters such as transmission times, GOES channel numbers, DCP address, etc.
DecodingScript	A structured scripting language that tells DECODES how to extract time-tagged samples from the raw messages
EU-Converters	How to convert raw values into engineering units, and how to convert between various types of engineering units (e.g. feet to meters).
PresentationInfo	How to format each type of sample. For example, you might want all stage values to be presented in centimeters with 10.3 resolution.
DataTypes	DECODES knows how to convert between the USGS (EPA) numeric codes and SHEF physical element codes.
Formatters	How to format data on output. Formatters are implemented for SHEF, SHEFIT, Human-Readable, and "Dump".
Consumers	Where to put the output data: files, directories, pipes, etc.
RoutingSpec	Puts all of the above together. A RoutingSpec says where to get the raw data, how to decode it, how to EU convert it, how to format it, and where to send it.

DECODES is written in 100% pure Java. Therefore there should be (almost) no porting issues in running it on any modern computing platform. Ilex Engineering has tested it under Windows 2000 and Linux.

DECODES merges the capabilities of the former DECODES software used by U.S. Geological Survey (USGS) with the EMIT (Environmental Message Interpreter Translator) software used by several U.S. Army Corps of Engineers (USACE) districts.

DECODES is open-source software. It was developed by Ilex Engineering, Inc., under a contract jointly funded by USGS and USACE. To obtain a copy of DECODES software, contact the U.S.G.S. Water Resources Division Headquarters.

For more information on Ilex Engineering, Inc., visit our web site at www.ilexeng.com,

call us at 410.465.6948, or email us at info@ilexeng.com.

2. DECODES General Concepts

Figure 2-1 provides a more detailed data flow diagram of what happens when data is decoded. In each box in the figure, different algorithms and parameters are applied according to information in your database.

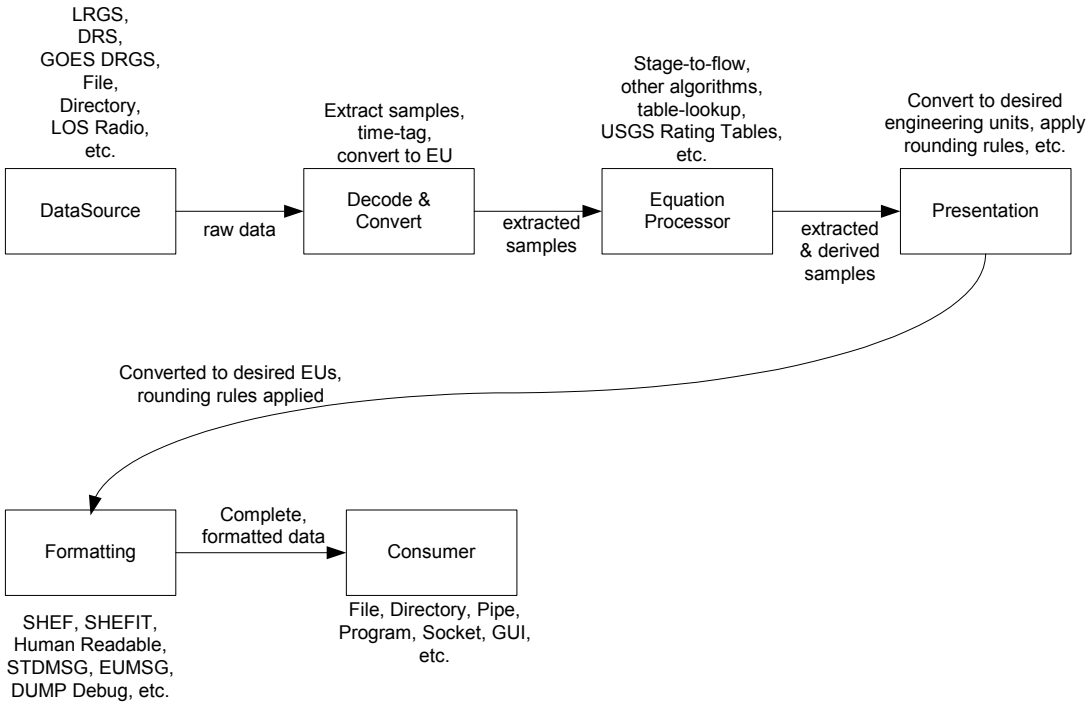

Figure 2-1: DECODES Detailed Data Flow

Figure 2-2 shows the DECODES software components and how they relate.

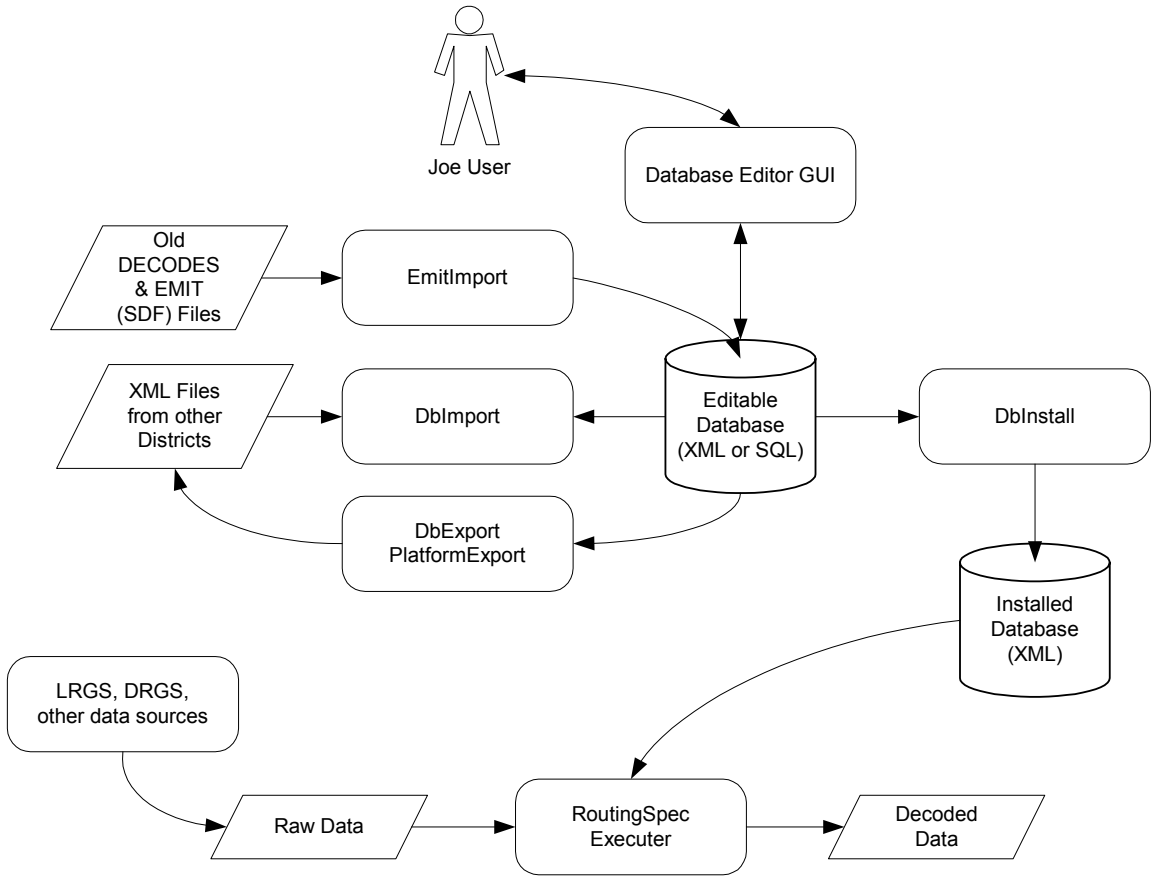

Figure 2-2: DECODES Components.

EmitImport can accept files from EMIT and legacy DECODES systems. These older software packages could export what was commonly called an SDF or Site Device File. If you currently use EMIT or an older version of DECODES, you can import your platform specifications directly into DECODES 5.

DbImport accepts XML files from other organizations using DECODES. A primary goal of DECODES is to encourage interagency cooperation.

Two utilities, DbExport and PlatformExport can be used to create XML files for exchange.

An extensive GUI database editor is provided for creating new DECODES specifications and modifying existing ones. The editor has features for interactively decoding raw data on-the-fly as you modify your specifications.

The DbInstall utility takes records from your editable database that you have blessed, and places them into the "installed database" for your production system.

The Routing Specs use information from the installed database to decode and convert raw data from a variety of sources.

3. DECODES Database Architecture

This chapter provides an overview of the DECODES database. For a more complete listing of elements, along with SQL and XML schema, see the DECODES 5 Database Schema Document.

We divide database records into two categories:

- **Setup Information** – These are records that should rarely change, such as standard unit conversions, data type records, etc.

Decoding & Converting Specifications – You will modify these records as you add, delete, or modify platforms; pull data from different sources; integrate new back-end databases; etc. An extensive GUI editor (see Chapter 7) is provided for maintaining these records.

Database information is shown using Entity Relationship Diagrams (ERD). These diagrams show the information contained in each entity and how different entities relate (shared keys, etc.).

3.1 Setup Information

Setup information should rarely change. The DECODES distribution comes with fully-populated tables of setup information. This will be sufficient for most organizations.

Currently the only way to modify Setup Information is to modify XML files by hand. Future releases will include a setup GUI.

3.1.1 Enumerations

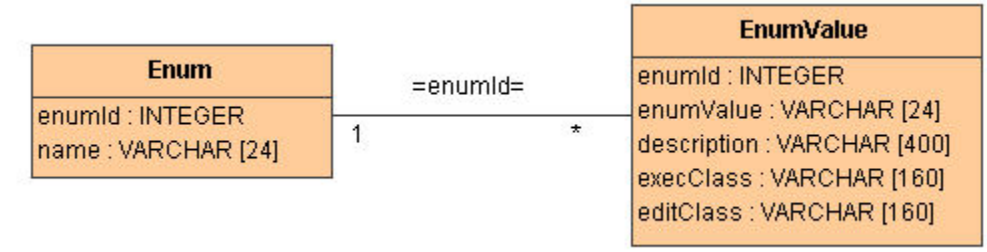

Figure 3-1: Enumerations ERD.

There are several places where an object must hold a string that must be constrained to one of several valid choices (i.e. an enumeration type). DECODES contains static tables in the database to store the valid choices for these enumerations. GUI programs can use these tables to offer pull-down menus for selection. They can also be used for validation when importing an XML file from another agency.

An enumeration is made up of a single 'Enum' entity and a series of associated 'EnumValue' entities. Think of 'Enum' as a type, and 'EnumValue' as the associated values that are valid for that type.

The Enum entity holds the name of the type. Table 3-1 lists the enumerations used in the DECODES database.

Enum Name	***Description***
SiteNameType	The 'type' values for site names. DECODES comes pre-loaded with four site name types: NWSHB5, USGS (site number), USGS-DRGS, Local.
DataTypeStandard	SHEF-PE, EPA-Code, etc.
RecordingMode	F=Fixed regular interval, V=Variable, triggered, or random.
ScriptType	Different ways to extract data from the raw platform messages. "Standard" scripts use the DECODES format statements to extract samples from your messages. This enumeration provides a hook for creating custom decoders.
TransportMediumType	GOES (either self-timed or random), File, Modem, NWSTG, etc.
DataOrder	A=ascending, D=descending. "Ascending" means that the earliest samples appear in the message first.
UnitFamily	"Metric" or "English"
UnitConversionAlgorithm	Four algorithms for converting from one EU to another: • "none" means no-conversion-necessary. In other words, the input and output units are synonyms. Example mL and cc. • "linear": $y = Ax + B$ • "USGS-Standard": $y = A * (B + x)^C + D$ • "Poly-5": $y = Ax^5 + Bx^4 + Cx^3 + Dx^2 + Ex + F$ Other algorithms can be easily added in the future.
Measures	A list of physical quantities that are measured by sensors. Used to associate units in different unit families. For example Meters and yards are related because they both are 'measures' of length. This list includes "arc", "area", "flow", "length", "temperature", "time", "velocity", "voltage", and "volume".
EquationScope	DCP, DCF, NL, SITE, ALL
LookupAlgorithm	Linear, Exponential, Logarithmic, Truncating, Rounding, or "Exact-Match"
OutputFormat	The following output formats are currently implemented • SHEF – Standard Hydrometeorologic Exchange Format • SHEFIT – Intermediate format used by USACE • Human-Readable – compact row/column format • EMIT-ASCII – Compatible with the "ASCII" format produced by EMIT • EMIT-ORACLE – Compatible with the "ORACLE" format produced by EMIT
DataConsumer	Data Consumers specify where to send data once it has been decoded, converted, and formatted: • File – write to a specified output file • Directory – write each message to a separate file in a specified directory. • Pipe – send data to standard output, usually for piping into another program.
DataSourceType	Data Sources provide raw messages to the decoder: • File – read raw messages stored in a file • Directory – Each file in the specified directory should contain a single raw message • LRGS – Connect to an LRGS or DRS over the network and pull messages • HotBackupGroup – Used to specify a group of LRGS systems. If one connection fails, use another in the group. • RoundRobinGroup – Read data continually from a group of other data sources (directories, files, LRGS, etc.) • SocketStream - Reads a stream of messages from a TCP socket.

Table 3-1: Enumerations in the DECODES Database.

3.1.2 Time Zones

The initial release 5.0 of DECODES included database entities for time-zones. These have been removed in DECODES 5.1 (and later).

DECODES now uses the internal Java time zone definitions. Routing Spec and Site entities hold references to time zones by storing a "TimeZone ID" recognized by Java.

A list of the time zones supported by Sun Microsystem's Java 1.4 is provided in Appendix B. You can also construct a custom time zone by specifying an offset to GMT. For example, "GMT-06:00" would mean 6 hours behind GMT, corresponding to Central Standard Time with no support for Daylight time.

3.1.3 Sensor Data Types

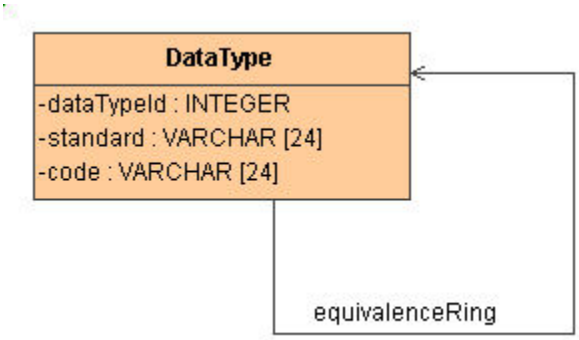

Figure 3-2: Data Types ERD.

DECODES can accommodate different systems for representing data types. Each data type is denoted by a 'standard' and a 'code'. For example "SHEF:HG" could be used for stream stage values.

The database also contains records that assert an equivalence between two data types. For example, the SHEF code HG is equivalent to the EPA code 00065. These records allow the software output data in different coding standards, regardless of the agency maintaining the DCP.

For example, USGS may prepare platform records using EPA codes. USACE could use these records without modification, telling DECODES to convert all the data types to SHEF.

3.1.4 Engineering Units

The DECODES database contains a list of commonly-used of standard and English engineering units. It also contains records that specify conversions between them. Figure 3-3 shows these database entries.

Conversions are performed with one of the following algorithms:

- None (means that EUs are synonyms like cc and ml)
- Linear
- USGS Standard Equation

5th order polynomial.

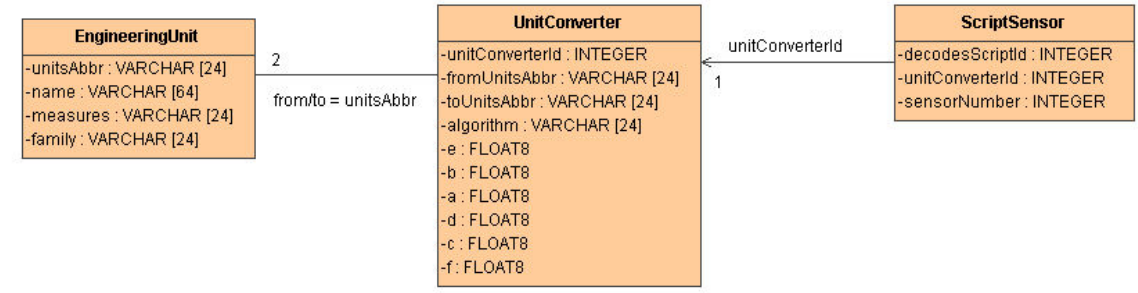

Figure 3-3: Engineering Units & Converters ERD.

3.2 Properties

There are several places in the model where we left “hooks” for arbitrary information that may be used by a particular agency. This type of information is stored in a set of properties associated with some other entity.

3.3 Decoding, Formatting, and Converting Specifications

3.3.1 Equipment Models

The EquipmentModel entity captures information about a piece of hardware, such as a platform, a transport medium, or a sensor.

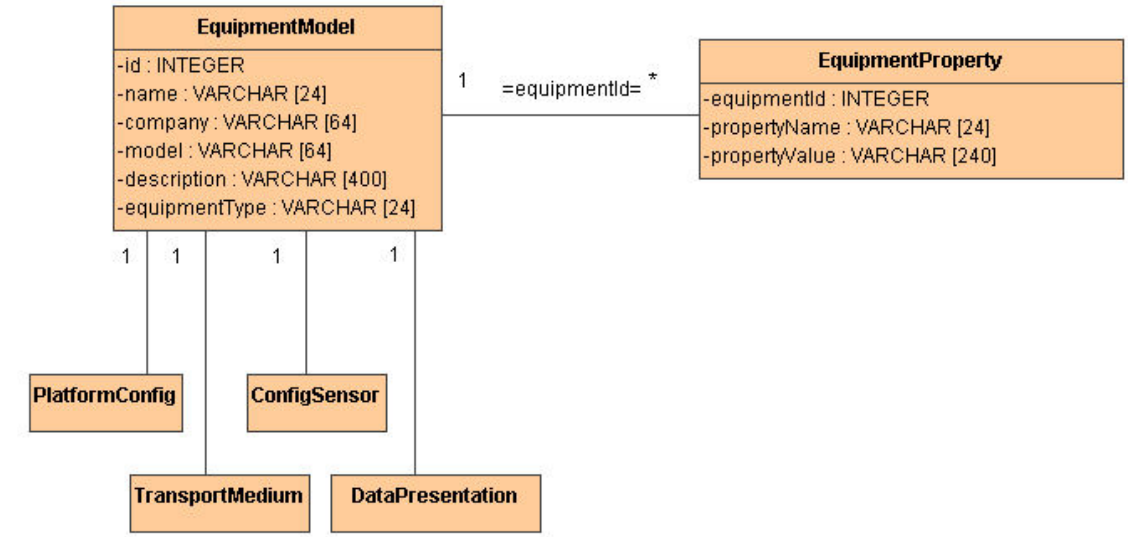

Figure 3-4: Equipment Model ERD.

The EquipmentModel entity stores information about the model and manufacturer of these pieces of equipment.

An equipment model may have an arbitrary set of properties. An example of a likely property might be time-ordering for platforms.

3.4 Sites & Site Names

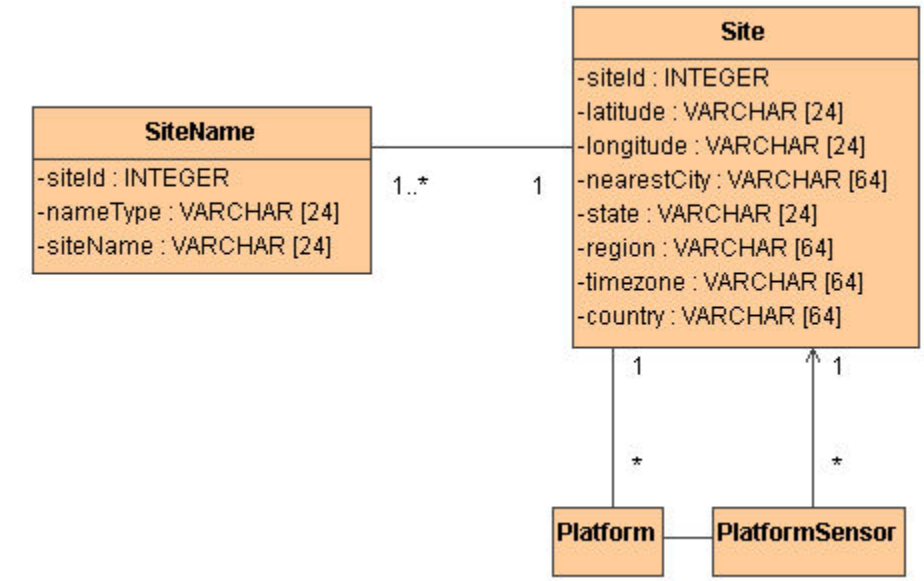

Figure 3-5: Sites and Site Names ERD.

In DECODES, a "Site" is simply a location. This is a little bit different from the concept in EMIT where a site is synonymous with a platform. In DECODES, a site may contain more than one platform, and a platform may be reporting sensors at different sites.

Consequently, the Site entity contains attributes that describe the location only.

A site can have many names. For example, USACE typically uses the National Weather Service HB5 name. USGS uses a numeric site ID. Other agencies may define a "local" name specific to their organization.

One or more sensors can exist at a site. Normally these are associated directly with the platform at the same site. However, sometimes a sensor can be associated with a platform at a different site.

3.5 Platform Information

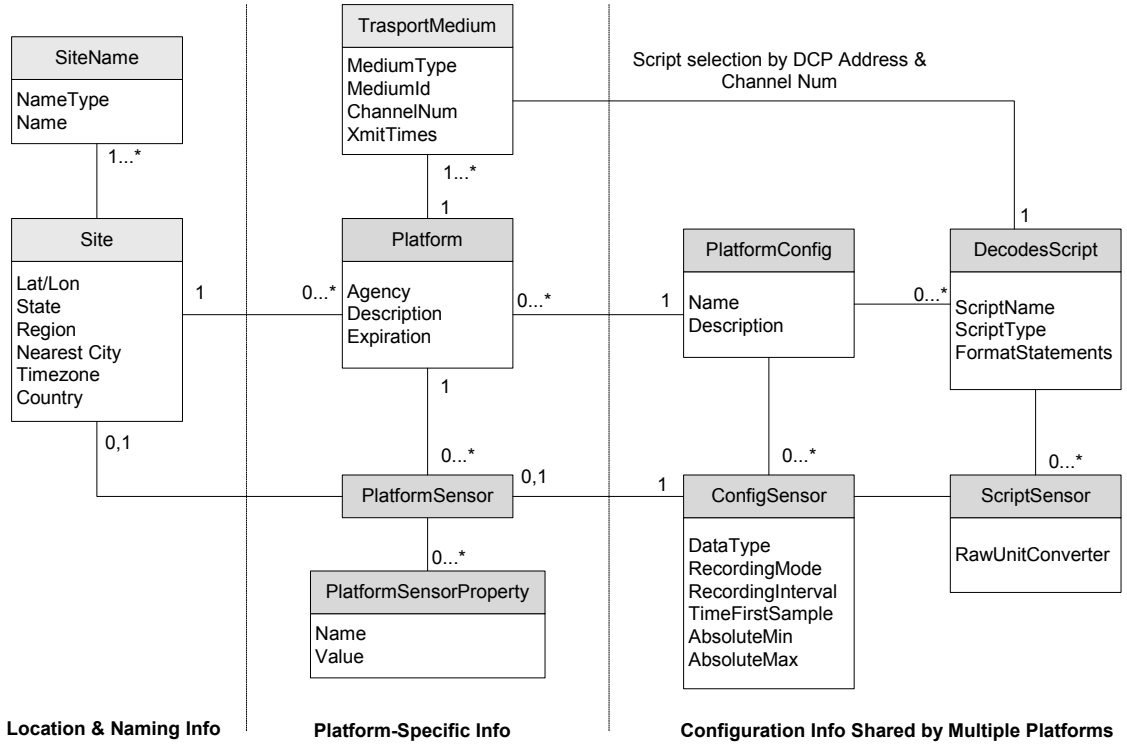

Figure 3-6: Platform Information ERD.

The "Platform" entity is the central entry point for decoding. The decoder recognizes an incoming message by its TransportMedium information. For example a GOES DCP has a unique DCP address. Each TransportMedium record points to a particular "Platform".

The Platform has a PlatformConfig which determines the sensors installed in the platform, and the scripts used to decode information from that platform.

Each PlatformConfig has a series of ConfigSensors. Each ConfigSensor has a unique "sensorNumber". Each ConfigSensor has a data type and may have a group of properties (ConfigSensorProperties).

The same PlatformConfig may be used by several platforms. For example, a group of Sutron 8200s which have identical sensors and message formats may share the same PlatformConfig.

Platform-specific information about sensors may be stored in the PlatformSensor and PlatformSensorProperty entities.

A DecodesScript contains the instructions for decoding messages from a platform that were received over a given transport medium. For example, a platform may have a data logger and a GOES transmitter. The GOES DCP messages would be decoded with one script, and the data logger files decoded with a different one. The choice of which script to use is based on which "TransportMedium" the data was received on.

3.6 Performance Measurements

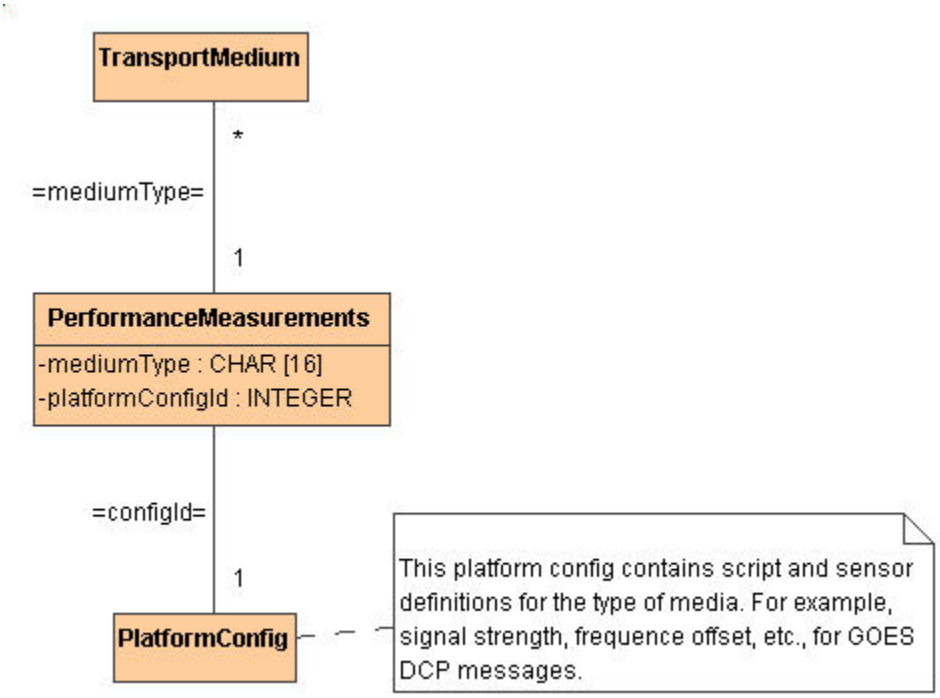

Figure 3-7: Performance Measurements ERD.

"Performance Measurments" contain information describing the message transmission. For example, a GOES DCP message contains:

- A time stamp
- GOES Channel number
- Message type indication
- Quality indicator (good or parity errors)
- Signal string

Carrier offset

This information is available to the decoder through a special PlatformConfig entity that is shared by all Transport Media of a given type. For example, there would be one config for all GOES DCP messages. This config would contain the "sensor" value (listed above) and decoding instructions for the message header.

3.7 Presentation and Unit Conversion

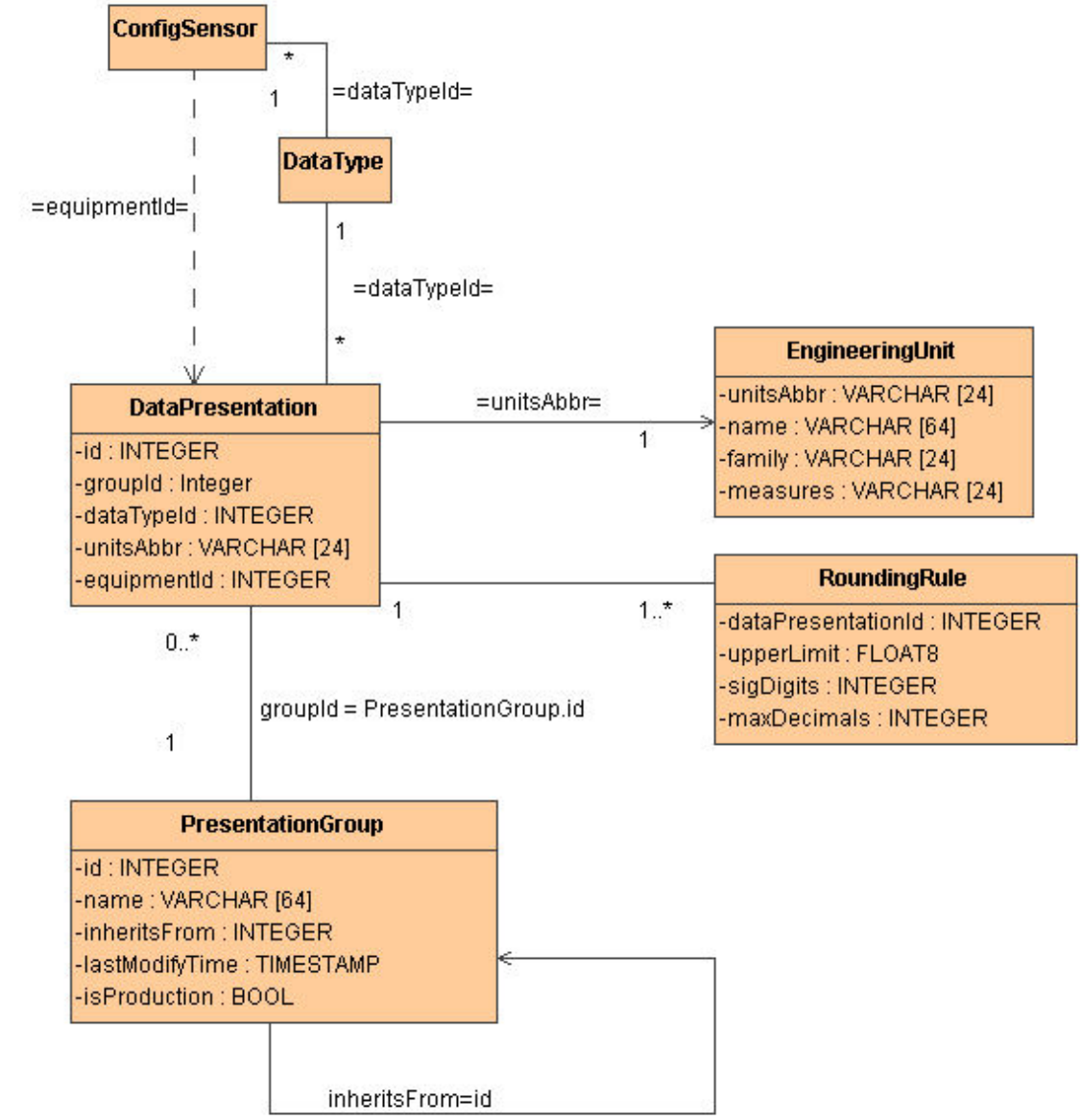

Figure 3-8: Presentation and Unit Conversion ERD.

Refer back to the diagram in the Platform Information section. Note that the "ScriptSensor" points to a "UnitConverter". In that case, the UnitConverter translates the raw value contained in a message into its initial Engineering Units value. A typical case would be a stage sensor that reports in tenths of inches. The initial unit converter would convert the value to inches by dividing by 10.

In Figure 3-8, a UnitConverter converts from one EU into another. A library of standard unit conversions is built into DECODES. This enables you to specify which units you want to output, regardless of who created the decoding specification.

An “EngineeringUnit” belongs to a family (English or metric). It also has a unique abbreviation (e.g. “mm”) and a full name (“millimeters”). It uses one of the algorithms listed in Table 3-1.

A “DataPresentation” entity also uses a set of RoundingRules to determine the display resolution for a given data type. For example, a DataPresentation entity for stage values might assert that values should be output in inches. and…

- If the value is between 0…1, use 3 decimal places
- If the value is between 1 and 10, use 2 decimal places

If the value is above 10, use 0 decimal places.

3.8 Equation Processor Information

Release 5.2 of DECODES does not include executable software for the Equation Processor. However, the database entities for equation processing and table lookup have been designed and are presented here.

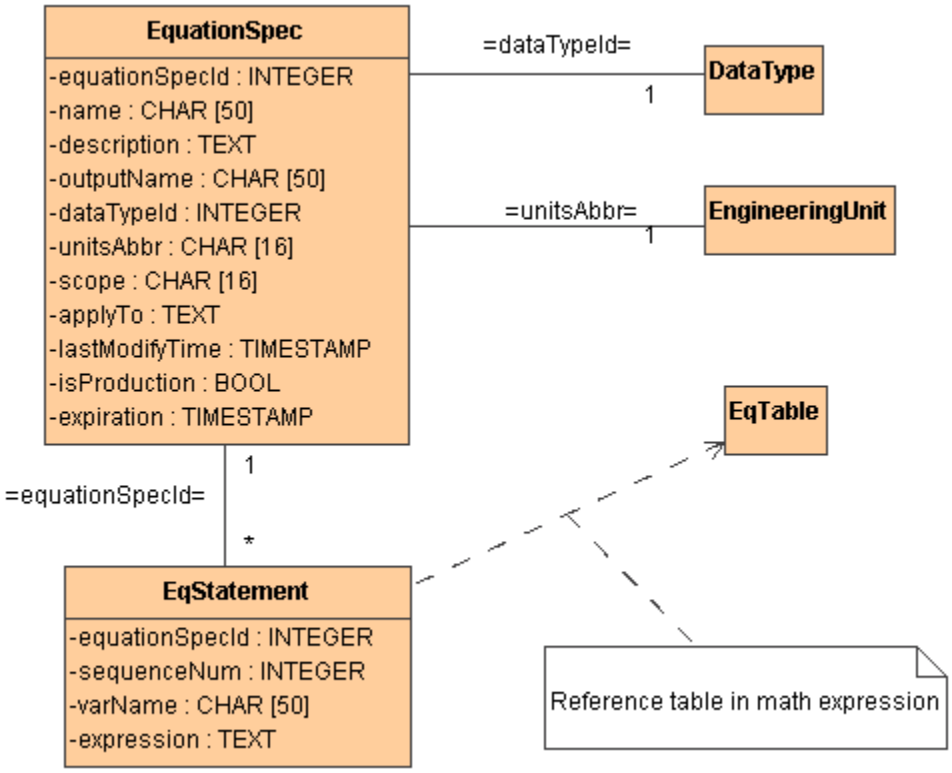

Figure 3-9: Equation Processor ERD.

The "Equation Processor" that can use arbitrary mathematical expressions to manipulate the data before it is output. It can be used to modify the output value or to create new derived output values.

An "EquationSpec" entity executes a series of EqStatements. Statements reference input variables by sensor name or by parameter codes.

A statement may include a call to a lookup function, thereby referencing an EqTable. A common use for this is in stage-to-flow conversions.

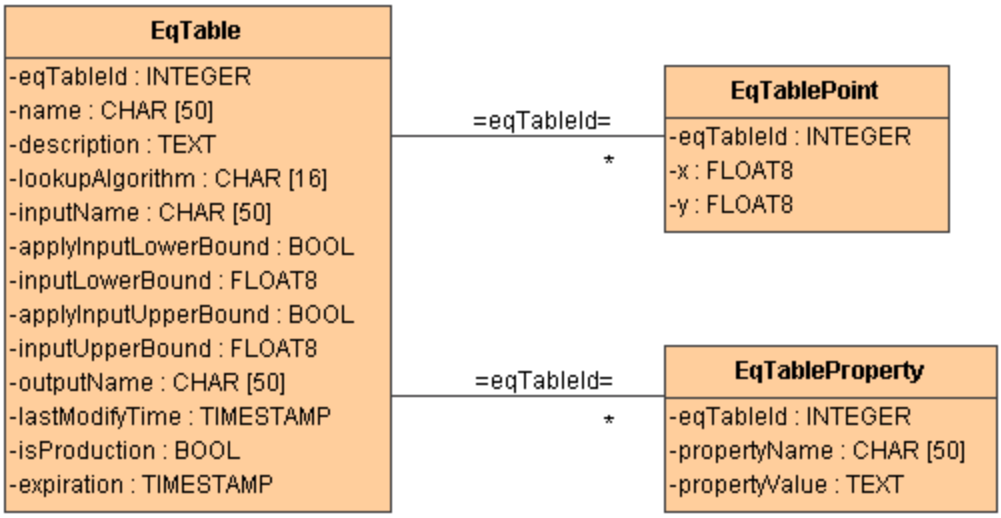

Figure 3-10: Lookup Tables ERD.

A lookup table is takes an input value and looks up an output value. A lookup table has a series of points ('x' is input, 'y' is output).

The 'lookupAlgorithm' attribute specifies how the lookup is done:

- Linear interpolation
- Logarithmic interpolation
- Truncating

Rounding

3.9 Routing Specification Information

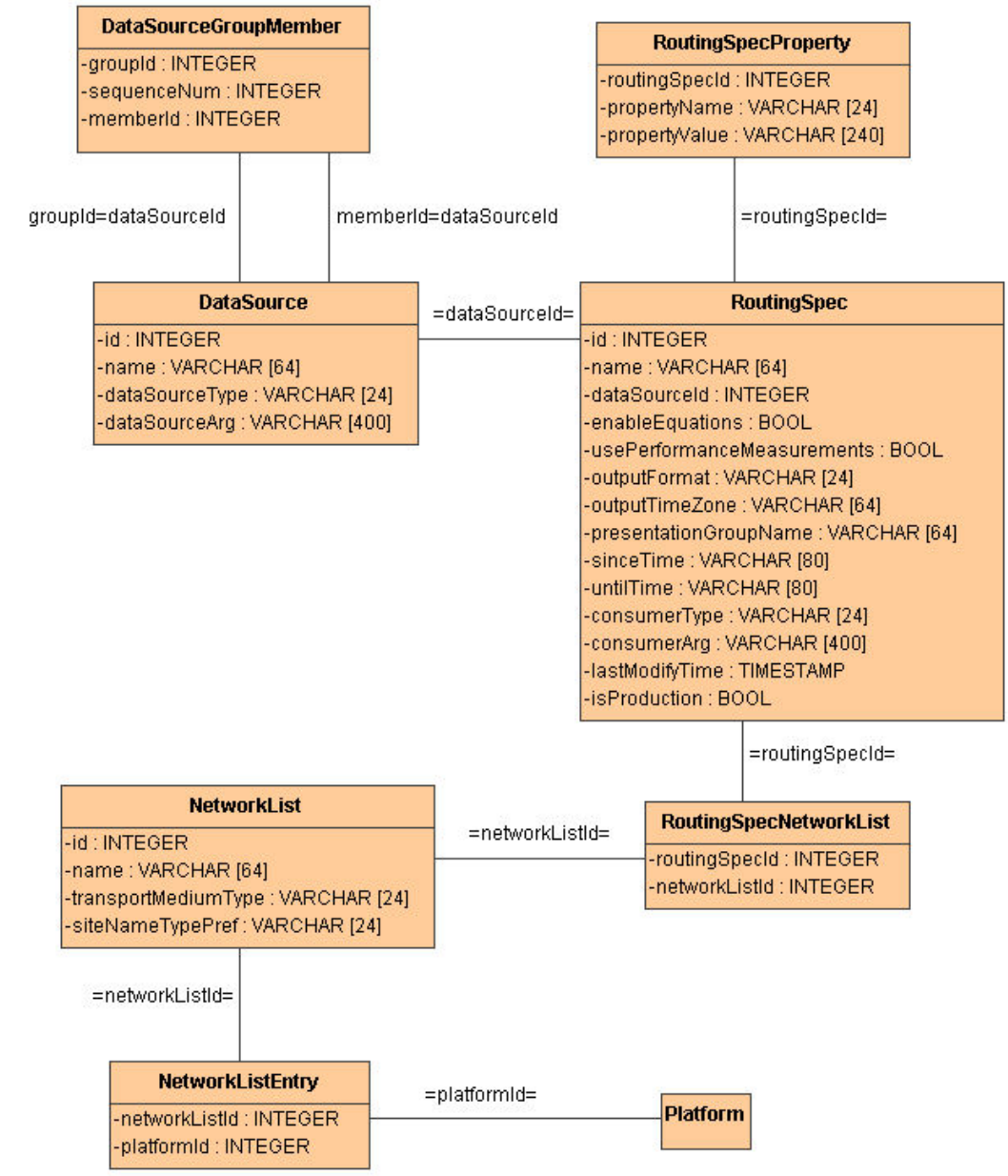

Figure 3-11: Routing Specification ERD.

DECODES uses a "Routing Specification" to determine:

- Where to get data from (Data source entities)
- Which data to get (Network Lists)
- How to format data for output

Where to send it once it is decoded and converted

A "NetworkList" in DECODES is analogous to a network list used by DOMSAT systems. It is simply a list of transport media (i.e. DCP addresses or NESS IDs).

4. Installing or Upgrading DECODES

There are two types of DECODES installations:

1. A New Installation – if you're installing DECODES to a particular computer for the first time, follow the instructions in sections 4.1 to install Java, then 4.2 to install DECODES.
2. An Upgrade Installation for sites that are running a previous version of DECODES. Consider upgrading to the latest version of Java, as described in section 4.1. Then do the DECODES upgrade as described in section 4.3

4.1 Installing Java

DECODES software is made up of Java archives and scripts for various operating systems. To run the Java code you will need to install Sun Microsystem's Java Runtime Environment version 1.3.1 or higher. This is available as a free download from:

```
http://www.javasoft.com
```

Download the "J2SE" Standard Edition. The latest stable version at the time of this writing is 1.4.1_03. You may download the "JRE", or if you are interested in doing Java development, you can download the "SDK" (Software Development Kit), which contains the JRE plus several development tools.

4.1.1 Installing Java on Windows

Follow these instructions to install the SDK on Windows 2000, NT or XP:

- Download the SDK release, as described above. The current version file name is j2sdk-1_4_1_03-windows-i386.exe. Download the file to your desktop or a temporary directory on your hard disk.
- Double click the icon to start the installation procedure.
- Read and agree to the Sun Microsystems License.
- Choose a destination folder for the Java SDK, or accept the default shown.
- Complete the release via the dialogs.

After installation, open a DOS window and type the command:

```
java -version
```

You should see a version message matching the release that you installed. If you see a message that 'java' is not recognized as an internal or external command, etc; then Java is NOT installed properly. Review the installation instructions above.

4.1.2 Installing Java under Red Hat Linux

If you want to run DECODES on a machine that is already set up as an LRGS, Java should already be installed. If not, download the JDK for Linux from the Javasoft website, it will be stored in a file named:

```
j2sdk-1_4_0_03-linux-i586-rpm.bin
```

Note -- Release 1.4.0_01 is current at the time of this writing -- the release numbers may be different by the time you read this. Make a note of the release you download and make substitutions to the file and directory names in these instructions.

Login as 'root' and move this file to the /root directory. Then run the shell script to unpack the RPM (RedHat Package Manager) file. Finally, install the RPM.

```
mv j2sdk-1_4_0_03-linux-i586-rpm.bin /root
cd /root
sh j2sdk-1_4_0_03-linux-i586-rpm.bin
(answer the questions about licensing agreement)
rpm j2sdk-1_4_0_03-fcs-linux-i386.rpm
```

This will result in the Java release installed in the following directory.

```
/usr/java/j2sdk1.4.0_03
```

We recommend that you set up a symbolic link pointing to this directory called /usr/java/jdk. You can do this as follows (note, you must be root to do this):

```
cd /usr/java
ln -s j2sdk1.4.0_01 jdk
```

Configure your Login Account for Java

You need to place the bin directory in the Java release into your PATH variable:

```
export PATH=/usr/java/jdk/bin:$PATH
```

Place this command in your .bash_rc file to have it done every time you login.

Verify that the path is properly set by typing:

```
java -version
```

4.2 Installing The Complete DECODES Package

The complete DECODES install is found in a ZIP file with a name of the form:

```
decodes-VERSION.zip
```

For example, the latest release is 5.4, so the ZIP file name is decodes-5.4.zip.

Download the zip file. You can find the latest version of DECODES on the "Download" page at http://www.ilexeng.com.

The complete DECODES Release contains:

- This document in PDF form
- A 'bin' directory containing executable scripts and Java Archive (JAR) files.
- An 'edit-db' directory hierarchy containing the setup information and a set of sample data files.
- An empty 'installed-db' directory
- A directory called 'sample-data' containing sample DCP messages that you can use to interactively decode inside the Database Editor GUI
- A directory called 'to_import' containing EMIT SDF and Network List files from 5 different USGS and USACE districts. These files have been used to test DECODES. You can import this data into your edit database while you are learning DECODES.

An 'sql-samples' directory containing setup scripts and other files to get you started using DECODES with a relational database (see Chapter 5).

The Windows-specific release also contains the JRE for Windows.

Create a directory for the DECODES installation. This will be called the "DECODES_INSTALL_DIR" in subsequent manual sections. Extract the Zip file there.

Unix Example: Suppose you chose to install the DECODES software under /usr/local/decodes, then you would enter the following at your shell prompt:

```
cd /usr/local
mkdir decodes
cd decodes
unzip decodes-5.2.zip  .
```

Windows Example: Suppose you chose to install the DECODES software under C:\decodes. After creating this directory run WinZip and extract the contents of the release into this directory.

CAUTION! Some versions of the unzip program, most notably the one included with XP, will try to create an intermediate directory with the same name as the zip file. **YOU DO NOT WANT THIS**. Check that you see the above listed files directory under the install directory.

4.2.1 Setup DECODES Runtime Environment

You need to create on new environment variable on your system:

DECODES_INSTALL_DIR - should be a hard-coded path to the directory where you unzipped the release.

You also need to add the bin directory under DECODES to your PATH variable.

On UNIX systems: Modify your startup script, such as .profile, .bash_profile, etc.

On Windows Systems: From the Start menu, select Settings - Control Panel. Double click on "System". Click on the "Advanced" tab and push the button labeled "Environment Variables…". You see a dialog as shown in Figure 4-1. Note the settings for DECODES_INSTALL_DIR and Path.

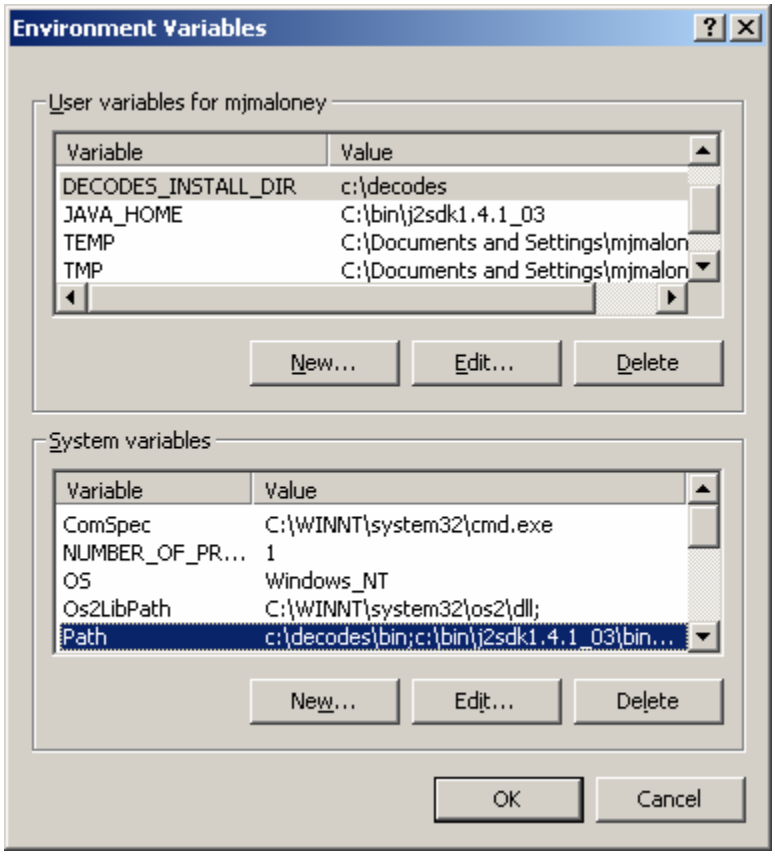

Figure 4-1: Windows 2000 Environment Variable Dialog.

Previous releases of DECODES used a file called "decodes-env" or "decodes-env.bat" that can be found under the DECODES release. If you set the two variables as described above, you no longer need these files.

4.2.2 Setup your DECODES Properties

Also contained in the *DECODES_INSTALL_DIR* directory is a file called decodes.properties. This file is a standard Java properties file (name=value pairs, one per line). The valid settings are shown in Table 4-1. "Default Value" is the value that will be used by the software if the property is missing from the file.

The DECODES distribution includes a sample decodes.properties file for use with the sample XML database (as the "editable" database). You should be able to get started using DECODES with this sample properties file; it's also located in the *DECODES_INSTALL_DIR* directory.

Property Name	***Default Value***	***Description***
DatabaseType	"xml"	This is the type for the installed database. The value should be either "XML" or "SQL". In the future, we hope to add "XMLURL", for an XML database accessible via the Internet.
DatabaseLocation	N/A	This is the location for accessing the installed database. For "XML"-type databases, this is a directory name containing the DECODES directory tree. For "SQL"-type databases, this is a JDBC database-url. See section 5.2 for more information about the format of this value. For "XMLURL"-type databases, this is a URL.
SiteNameTypePreference	NWSHB5	Specifies the "preferred" type for DCP names. By default this is the Handbook-5 standard used by the National Weather Service.
EditDatabaseType	"xml"	The is the type for the Editable database. The same values allowed for the DatabaseType property are allowed here; viz "XML" or "SQL".
EditDatabaseLocation	N/A	This is the location for accessing the editable database. The same values allowed for the DatabaseLocation property are allowed here.
EditOutputFormat	"Human-Readable"	Output format used to test decoding scripts within the editor.
EditPresentationGroup	N/A	Presentation group used to format samples when testing decoding scripts within the editor.
EditTimeZone	"UTC"	Time zone used when decoding sample data within the editor. Using UTC makes it easy to correlate sample times with the DCP message time stamp.
jdbcDriverClass	org.postgresql.Driver	If you use a SQL database other than PostgreSQL, specify the full Java class name of the JDBC driver here.

Table 4-1: DECODES Property Values.

4.3 Upgrade Installation

If you are upgrading from an earlier version, download the DECODES upgrade Zip file from the Ilex web site. The upgrade Zip file will have a name like

```
decodes-5.4-upgrade.zip
```

This Zip file contains:

- This document in PDF form
- A 'bin' directory containing executable scripts and updated Java Archive (JAR) files.
- An 'edit-db' directory hierarchy containing the latest version of the setup XML files, as described above (see section 3.1).
- An 'sql-samples' directory containing setup scripts and other files to get you started using DECODES with a relational database (see Chapter 5 below).

To install the upgrade, simple unzip the file over the same location that DECODES was previously installed.

Note that the upgrade does *not* contain the environment file, the properties file or the initial database files. Therefore it is safe to install the upgrade without fear of losing your existing DCP configurations.

5. Using DECODES with an SQL Database

As of version 5.3 of DECODES, The preferred medium for your editable database is SQL. The advantage is that SQL databases are accessible over the network. Hence you can run your database editor and routing specs locally, on a windows PC, and access a SQL database that resides on a central office server.

Although DECODES should work with any RDBMS that supports JDBC, at the time of this writing, it has only been tested with PostgreSQL, a free DBMS that runs on most operating systems, and is included in the Red Hat Linux Release.

5.1 PostgreSQL

PostgreSQL was chosen for our initial implementation for many reasons:

- It's free.
- It runs on a wide variety of different machines. One of the key features of DECODES itself is that it is very platform-independent. By choosing PostgreSQL, we continue that tradition.
- It is extremely well documented.

It's powerful, mature, and scalable.

This section will give a brief description of how to setup PostgreSQL on either a Linux or a Windows computer. For other types of computers, or for more detailed instructions, consult the documentation that comes with PostgreSQL. Two excellent sources of information are:

- The PostgreSQL web site: http://www.postgresql.org/, and

Practical PostgreSQL; John C. Worsley and Joshua D. Drake, O'Reilly & Associates, Inc, 2002.

5.1.1 Installing PostgreSQL

You may not need to install PostgreSQL. If you are using Linux, then PostgreSQL may already installed on your system. Try entering the following command:

```
psql -V
```

If this responds with version information about the PostgreSQL, then you're in luck. If not, find the postgreSQL RPM packages in your Red Hat distribution or on the Red Hat web site and install them.

For systems other than Red Hat Linux and Windows, consult the detailed instructions in the Administrator's Guide, which is available online at

```
http://www.postgresql.org/idocs.
```

5.1.1.1 Installing PostgreSQL on Windows 2000, NT, or XP

There is an open-source package called "Cygwin" containing many Unix-like tools that can run under Windows. PostgreSQL is included in Cygwin. You can get Cygwin from:

```
http://www.cygwin.com/
```

Unfortunately however, the PostgreSQL implementation in Cygwin is, at the time of this writing, incomplete. Look inside the release (wherever you installed it) for the file usr/doc/postgresql-7.2.1/FAQ_MSWIN. This file contains additional instructions that you'll need to follow to get a PostgreSQL server running under windows. In particular, note the section on installing cygipc. You will need to go to the web site:

```
www.neuro.gatech.edu/users/cwilson/cygipc/index.html
```

Future releases of Cygwin may contain a fully-functional version of PostgreSQL, so check the release notes for details.

5.1.2 PostgreSQL Startup and Initialization

On Linux, PostgreSQL will create a default database cluster and environment the first time it is started. These instructions assume that you will use this default environment for the DECODES database.

If you want to use a different cluster, see the documentation on the 'initdb' command.

To start PostgreSQL manually, login as root and:

```
cd /etc/init.d
./postgresql start
```

To have PostgreSQL started automatically when the system boots:

```
chkconfig --level 2345 postgresql on
```

Verify that the server is running by starting a 'PSQL' session on the default "template1" database. Again, do this as root:

```
psql -U postgres template1
```

5.1.2.1 Start the PostgreSQL Server on Windows 2000 or NT

On NT or Windows 2000 systems you will first need to start the Cygwin IPC daemon. To do this, open a Cygwin window and type:

```
ipc-daemon &
```

After that, you can immediately start the PostgreSQL server with the command (again, entered as user postgres)

```
postmaster -i -D /home/postgres
```

If you picked a different directory location for your database cluster, use that as the argument to the "-D" option. This starts the server in a foreground process. You should see several response messages on the shell window. If everything is working, one of the last messages should be:

```
DEBUG:  database system is ready
```

5.1.3 Create PostgreSQL Administrative Account

The next step is to create a PostgreSQL user for DECODES database administration. This account will be used to create the DECODES database, and to create other user accounts. Note that this is different from a Unix (or Windows) user account.

We recommend that you create a user account called "decodes_adm". Some of the scripts (and this manual) assume that this user has been created and has complete permissions on the DECODES database.

Login as root, and execute the following commands

```
su - postgres
createuser -P decodes_adm
```

This will ask if you want to allow this user to be able to create databases and/or to create new users. Answer "yes" to both of these questions.

5.1.4 Setup PostgreSQL Security

PostgreSQL security is controlled by the file "pg_hba.conf" in the directory where your database is located. By default on a Linux system this will be in /var/lib/pgsql/data. This section describes how to establish simple password-based authentication. This is sufficient for a network that is secured from the internet by firewall(s). There are other ways of configuring authentication that are more secure. Consult the PostgreSQL documentation for details.

To allow database connections from users operating on the local machine add these lines to the pg_hba.conf file:

```
local all                                       password
host  decodesedit  127.0.0.1  255.255.255.255  password
```

The first line allows local connections to any database that use Unix domain sockets. The second line allows TCP connections to the 'decodesedit' database that come from the loopback device (127.0.0.1).

The following line will allow connections to all databases from the specific host (192.168.1.50):

```
host all 192.168.1.50  255.255.255.255 password
```

The following line will allow connections to the 'decodesedit' database from any host on the 192.168.1 network:

```
host decodesedit  192.168.2.0  255.255.255.0  password
```

You need to create PostgreSQL user accounts for all users that will edit the DECODES editable database. Use the 'createuser' command for this. These users do *not* need to be able to create databases or new users.

You will also need to configure the PostgreSQL server to answer incoming TCP connections. To do this modify the file /var/lib/pgsql/data/postgresql.conf. Add a line at the end as follows:

```
tcpip_socket = true
```

5.2 How DECODES Does Database Authentication

DECODES programs are written in Java. They use JDBC to access the SQL database. When a program starts, it needs to open a JDBC connection by supplying a username and password to the database server. The programs will look in a file called ".decodes.auth" in your home directory. This file contains your username and an encrypted version of your password.

You must create this file in your home directory and set the permissions so that only you can read or write it:

```
cd
touch .decodes.auth
chmod 600 .decodes.auth
```

Then run the 'setDecodesUser' script will place your database username and password into this file. For example, if your database account name is johnwarfen and your password is lectroid, type:

```
bash
setDecodesUser johnwarfen lectroid
^D
```

As described in the section above on PostgreSQL authentication, your database account is not the same as your Unix login account. Indeed, you could have several Unix users sharing the same database account for simplicity.

Once set, the username and password will remain in effect for you until you change it.

5.3 Configure DECODES for an SQL Editable Database

The "decodes.properties" file determines the type and location of your editable and operational (a.k.a. "installed") database. To switch to SQL, you will modify the properties "EditDatabaseType" and "EditDatabaseLocation".

Change EditDatabaseType to "sql". Change EditDatabaseLocation to a JDBC URL that specifies the name and host of your database. The syntax for the location URL would be:

```
jdbc:drivername:databasename
```

…where *drivername* is the name of the JDBC driver (e.g. "postgresql") and *databasename* is in one of the following forms:

- dbname For a local database on the default port.
- //host/dbname For a database on the specified host at the default port.

//host:port/dbname For a database on the specified host at the specified port.

Example: For a PostgreSQL editable database called "decodesedit" on host "mylrgs", place the following the decodes.properties file:

```
EditDatabaseType=sql
EditDatabaseLocation=jdbc:postgresql://mylrgs/decodesedit
```

For the editable database, we recommend using the database name "decodesedit". Some of the scripts, and this manual, assume that this is the case.

Also in the "decodes.properties" file, you will find a setting that specifies the full Java class name of the JDBC driver. The setting is called "jdbcDriverClass" and its default value is "org.postgresql.Driver". If you use a database other than PostgreSQL, determine the name of the driver class, and set this value as appropriate.

5.4 Creating the DECODES Editable Database

After you have configured database authentication (as described in section 5.1.4 for PostgreSQL, and run the setDecodesUser account (as described in section 5.2), and set your decodes.properties file (as described in section 5.3); then you are ready to create and initialize the DECODES database.

The script createDecodesDb.sh will do several things:

- Create a new database called "decodesedit".
- Use the SQL interpreter to define the tables in the database.

Runs the dbimport Java program to populate the database with enumerations, EU conversions, and data types.

The script has been written and tested using the PostgreSQL 'psql' program. If you are not using PostgreSQL, or if your database is not called 'decodesedit', you will need to modify the script accordingly.

The script initialized several tables from XML files found in the directory "./edit-db". For this reason you must run the script in the directory where you installed DECODES:

```
cd $DECODES_INSTALL_DIR
createDecodesDb.sh
```

You will see several messages printed to the screen as the script does its work. If any errors occur, descriptive information will be printed. If errors are encountered during the 'dbimport' phase of the script, additional information will be printed in the file "util.log".

If you are running the script a second time, you may see error messages from the create-database or table-definition phase saying that the database or table already exists. It is safe to ignore these warning messages.

If you really want to delete the database and start with a clean slate, for PostgreSQL, issue the command:

```
dropdb decodesedit
```

CAUTION: This command will delete the database and all data in it!

5.5 Importing Data from your old XML Database

If you already have an old XML database and you want to import your platforms, configs, etc., read this section.

The 'dbimport' script takes XML filename on the command line and imports them into your editable database. So, after following all the instructions in the previous section, you can issue any of the following commands.

```
# Go to the directory where the old XML editable DB was:
cd $DECODES_INSTALL_DIR/edit-db

# Import all my equipment model records:
dbimport equipment/*.xml

# Import all my Site records:
dbimport site/*.xml

# Import all my PlatformConfig records:
dbimport config/*.xml

# Import all my Platform records. Note: Don't import
# the file "PlatformList.xml"
dbimport platform/p*.xml

# Import all my DataSource records:
dbimport datasource/*.xml

# Import all my Presentation Groups Note: the init
# script will get "SHEF-English". You only need to do
# this if you have created other presentation groups.
dbimport presentation/*.xml

# Import all my network lists:
dbimport netlist/*.xml

# Import all my RoutingSpec records:
dbimport routing/*.xml
```

5.6 Verify that the Database Works

You should now be able to run “dbedit”. The editor should come up. You should see all of the data that you had previously defined in the XML database.

It should now be transparent to you that you are using a SQL database. All the scripts described in other sections of this manual should function normally:

- emitimport To import EMIT and Legacy DECODES SDF Files
- dbimport To import XML files from backups or other DECODES sites.
- pxport To export platform XML files
- dbedit To run the GUI Database Editor
- dbinstall To copy all “production” level entities to the installed database.
- markproduction To mark all entities in the editable database as “production” level.
- msgaccess To interactively view raw and decoded DCP messages
- nl2lrgs To extract a network list into an LRGS compatible ASCII file.
- rs To run a routing specification.

6. Maintaining the DECODES Database

Take a moment to refer back to Figure 2-2. Note that the Editable Database is separate from the Installed database. This section describes tools that you will use to:

- Import EMIT and pre-release-5 DECODES files into the editable database.
- Import Platform and other database XML files from other organizations using DECODES 5+.
- Create a new Editable database from scratch.
- Install components from the Editable Database to the Installed Database

6.1 Initializing Your Editable DECODES Database

This section describes how to initialize and populate your first editable database.

6.1.1 Standard XML Setup Files

If you are using an SQL database as your editable database, refer back to Chapter 5. This section is for the older XML editable database.

The setup files contain collections of information that probably does not need to change from one organization to another. Currently the only way to edit this information is to use a text editor to modify the XML files.

The files are all found under *$DECODES_INSTALL_DIR*/edit-db:

datatype/DataTypeEquivalenceList.xml: Described in section 3.1.3, this file contains definitions for common SHEF and USGS/EPA numeric type codes. You may want to edit this file if you use uncommon or custom type-codes.

enum/EnumList.xml: Described in section 3.1.1. You will probably not need to edit this file unless you are adding custom Java code to the DECODES system.

eu/EngineeringUnitList.xml: Described in section 3.1.4, this file contains records which describe most commonly-used English and Metric units. It also contains conversion algorithms & coefficients to convert between units that measure the same physical parameter.

6.1.2 Importing Data from EMIT or Pre-Release-5 DECODES

Synopsis:

```
emitimport <options> file1 file2 . . .
```

Options:

`-t name-type`	Sets the preferred site-name type to the specified value. The default is NWSHB5.
`-d debug-level`	Level should be 1, 2, or 3 from the least to most verbose.
`-v`	Validate only: Do not actually import any data. Just issue warnings about conflicts and parsing errors.
`-x`	Instead of importing into your editable-database, create an XML file containing the converted data. With this option, the program is a translator rather than an importer.
`-o`	Keep old records on conflict. Default is to overwrite old records with new ones.

Description:

This program can accept SDF files (SDF stands for Site Device File) from EMIT or pre-release-5 DECODES. It can also accept network list files from LRGS or DRS systems.

It creates new DECODES database records and places them into the editable database.

This program writes log messages to a file called “util.log” in the current directory.

Examples:

DECODES contains several sample files in the to_import sub-directory. To import these files type:

```
cd $DECODES_INSTALL_DIR
emitimport to_import/*
```

6.1.3 Importing XML Data from Other DECODES Sites

Synopsis:

```
dbimport <options> file1 file2 . . .
```

Options:

`-d debug-level`	Level should be 1, 2, or 3 from the least to most verbose.
`-v`	Validate only: Do not actually import any data. Just issue warnings about conflicts and parsing errors.
`-o`	Keep old records on conflict. Default is to overwrite old records with new ones.

Description:

This program accepts XML files that were created by the export utilities described in section 6.4. Imported records are added to your editable database.

This program writes log messages to a file called "util.log" in the current directory.

Examples:

```
pxport -a > platform-dump.xml
...at a different organization
dbimport platform-dump.xml
```

6.2 Interactively Editing the Database

The script 'dbedit' starts the interactive Database Editor GUI. See section 7 for instructions on using this program.

6.3 Updating the Installed Database

Synopsis:

```
dbinstall <options>
```

Options:

`-d debug-level` Level should be 1, 2, or 3 from the least to most verbose.

Description:

The purpose of this program is top copy records that are ready for production from your Editable database into your Installed database.

This program writes log messages to a file called “util.log” in the current directory.

The dbinstall program copies all of the setup information from your editable database into your installed database.

If you examine the diagrams for the various database entities in you will see “isProduction” parameters in the following record types:

- EqTable
- EquationSpec
- Platform
- PresentationGroup

RoutingSpec

For these record types, only the entites where the “isProduction” value is true will be copied. You can set this value for selected records in the Database Editor GUI (See section 7)

As a time-saving alternative, special script called ‘markproduction’ has been prepared. This script sets the isProduction flag to true for all records in the editable database.

Examples:

```
...after importing records to the editable database
dbinstall
```

6.4 Exporting Platforms to XML Files

Synopsis:

```
pxport <options>
```

Options:

`-d debug-level`	Level should be 1, 2, or 3 from the least to most verbose.
`-n network-list`	Export platforms referenced by the named network list.
`-s site-name`	Export the platform record for a specific site.
`-a`	Export all platforms.
`-c config-name`	Export platforms that use a given platform configuration.
`-i`	Export from the installed database. The default is to export from the editable database.

Description:

This program writes XML records containing platforms (and all subordinate records such as site, config, script, and transport media). Records are written to standard output.

Multiple instances of the above options are acceptable. See examples below.

This program writes log messages to a file called "util.log" in the current directory.

Examples:

Dump all platforms to a single XML file:

```
pxport –a > platform-dump.xml
```

Export three specific sites:

```
pxport –s TCLG1 –s HUDG1 –s LHMG1 > threesites.xml
```

Export platforms referenced by Atlanta's network list:

```
pxport –n Atlanta > Atlanta-platforms.xml
```

6.5 Other Database Utilities

6.5.1 Creating the Platform Cross-Reference File

If you look in the edit-db/platform directory you will see that each platform is stored in a separate file named with a 'p' followed by a numeric ID assigned to the platform.

Each database assigns an arbitrary numeric key ID field to platforms as they are added to the database. For the most part this key is invisible to you and you shouldn't have to worry about it.

Also in this directory is a file called PlatformList.xml. This file is a cross reference that maps site names, configuration names, and DCP addresses to each platform.

If you suspect that your cross reference file has been corrupted, you can rebuild the PlatformList.xml file directly. This can happen if you add files to the platform directory using tar, zip, cp, copy, etc.

To run the program type:

```
java decodes.xml.CreatePlatformXref  database-root
```

where 'database-root' is the path to the top of the XML database. For example, if you want to build the cross reference in your editable database, and you installed DECODES in /usr/local/decodes, type:

```
java decodes.xml.CreatePlatformXref /usr/local/decodes/edit-db
```

6.5.2 Creating LRGS-Style Network List Files

The LRGS and DRS support an alternative file format for network lists. LRGS Network Lists are ASCII files containing a list of DCP addresses, one per line:

```
DCP-Addr:DCP-Name  Comment…
```

The line starts with a (8 hex-digit) DCP address, followed by a colon, followed by a blank-delimited DCP Name, followed by a free-form comment field.

When you run a routing spec that uses an LrgsDataSource, the software converts any network lists you specified into the above format, and sends them to the server. The LRGS DCP Data Server (DDS) then only sends the messages from those platforms.

Older SCO-DRS servers do not support network list transfers. If your data source is a SCO DRS, do the following:

1. Add a property called "sendnl" set to the value "false" to the data source record.
2. Generate the LRGS-style network list file using the "nl2lrgs" utility (see below).
3. Use FTP or some other file-transfer mechanism to copy the list onto the SCO DRS. Place it in the 'drs' home directory (/usr/drs).
4. Repeate steps 2 and 3 every time the list is modified.

The "nl2lrgs" utility

```
nl2lrgs [-e] list1  list2  ...
```

The "nl2lrgs" will create a file in the current directory for each list specified. The file will have the same name as the network list, with a ".nl" extension.

The "-e" argument forces the utility to pull the network list from your editable database. The default is to use the installed database.

7. The DECODES Database Editor

Synopsis:

```
dbedit <options>
```

Options:

`-d debug-level` Level should be 1, 2, or 3 from the least to most verbose.

The editor starts as shown in Figure 7-1.

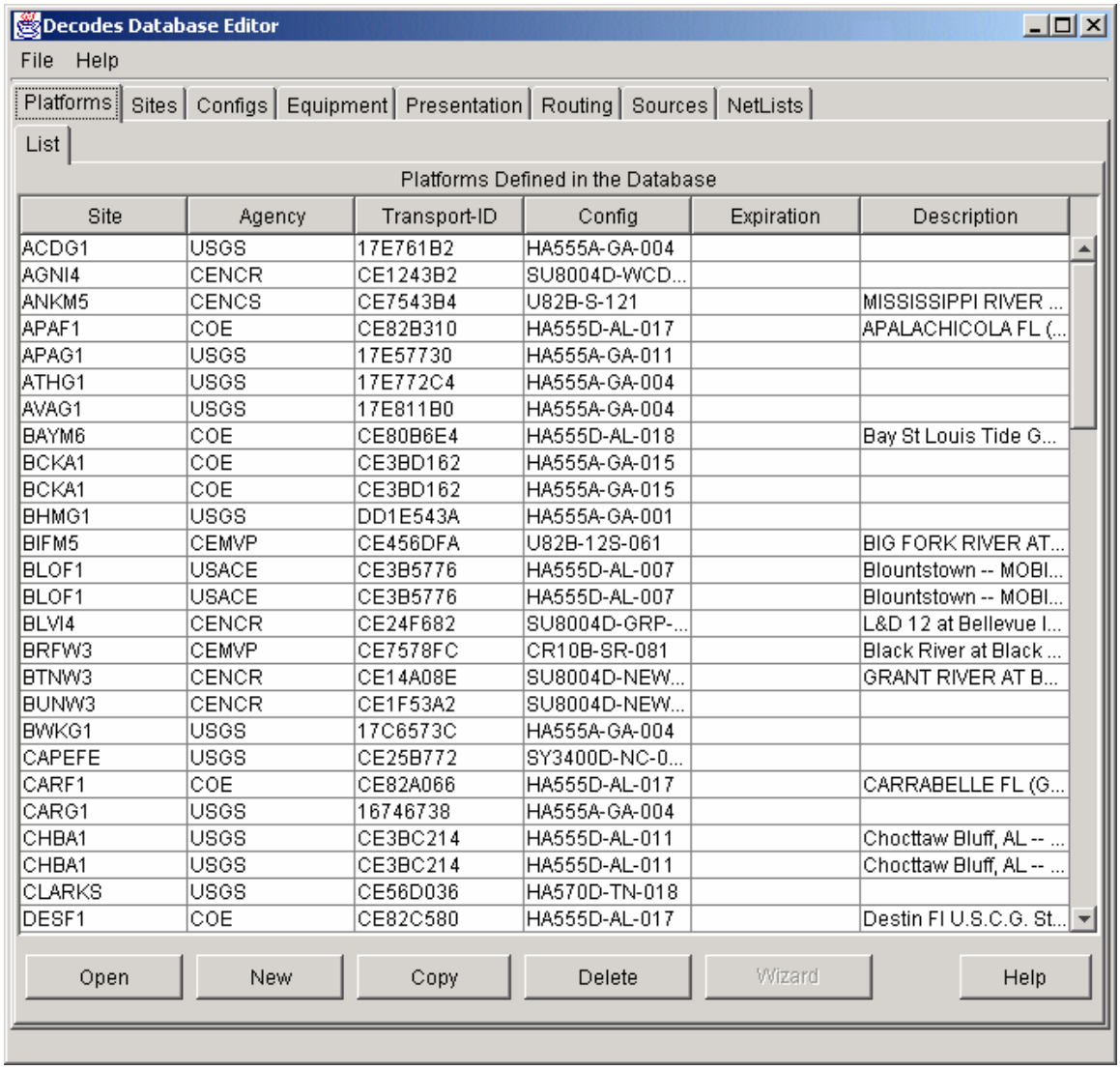

Figure 7-1: Database Editor Platform List Screen.

7.1 GUI Organization

A row of Tabs appears along the top corresponding to the different kinds of records in the database (Platforms, Sites, Configs, etc.)

7.1.1 List Panels in General

Underneath each of those tabs you will see a "List" tab. In Figure 7-1, the Platform tab is selected, so we see the List of platforms.

Click on the column header in the List tab to sort the elements by the columns value. In Figure 7-1 the 'Site' column header was clicked, so we see elements sorted by Site name.

Along the bottom of the List tab you see buttons with the following labels:

Open To edit a database record, click on it in the list and press Open

New Press new to create a new database record.

Copy To copy a database record, click on it in the list and press Copy. You will be prompted for a name for the copy.

Delete To delete a database record, click on it in the list and press Delete.

Wizard This is a placeholder for future features. We plan to implement wizard dialogs to guide you through creating various types of database entries. Currently the only 'Wizard' implemented is under the Configs tab. This Config wizard is simply a prototype that doesn't do anything currently.

Help Coming soon: This button will bring up a context sensitive help screen.

7.1.2 Edit Panels in General

When you **Open** a record, a new tab appears to the right of the list tab. For example, Figure 7-2 shows the result after we do the following:

- Select the **Configs** top-level tab.
- Select the record HA555A-GA-008 from the list

Press the **Open** button.

Separate edit screens have been implemented for each type of database record. In this edit screen you would change all of the parameters for HA555A-GA-008.

Notice the bottom of the Edit Panel. The **Commit** button writes the record back to the database. You can do this at any time. It does not close the panel.

The **Close** button closes the edit panel. If you have made changes to the record you will be asked if you want to save them.

7.1.3 Exiting the Editor

You can exit the editor by selecting File-Exit or by closing the window. If you have edit panels open in which changes have not been saved, you will be forced to close these panels before you can exit.

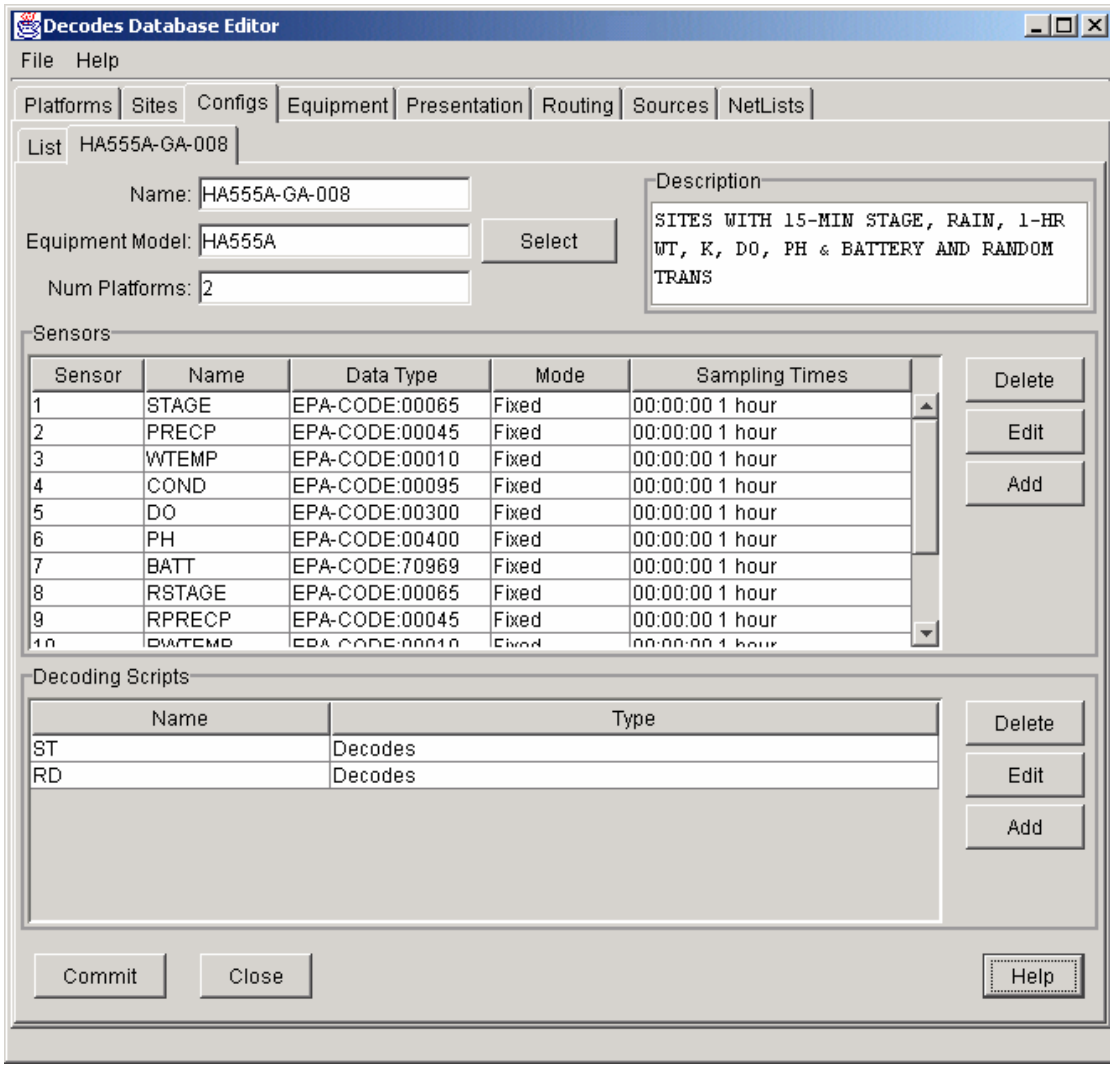

Figure 7-2: Platform Config Edit Panel.

7.2 The Platform Edit Panel

The platform edit panel is shown in Figure 7-3.

Every platform is associated with a site. Press the Site Choose button to bring up a dialog in which you can select a site.

Every platform is associated with a configuration. Press the Config Choose button to make this association. Be careful: Sensors are defined in the configuration.

The Owner Agency and Description are simple free-form type-in fields. They are informational and not used by other DECODES software.

If you want this platform to be placed into the 'install-database' by the dbinstall utility (see section 6.3), check the 'Production' checkbox.

Platforms change over time (sensors are added, removed, etc.). You can capture a historical version for a platform by pressing the 'Make Historical Version' button. Each historical version is a separate record with a specified Expiration time.

The Platform Sensor list is used for two purposes:

- If a sensor on this platform is actually located at a different site, you can associate the sensor with a site. In most cases, however, the "Actual Site" field is blank, meaning that this sensor is at the same site as the platform.

You can associate arbitrary properties with a sensor to be used by downstream DECODES modules. In the example shown, the USGS DBNO and DDNO are associated with each sensor.

Transport Media define how the data from this platform is retrieved. The data may need to be decoded differently depending on whether it was received over DOMSAT, DRGS, or EDL file, even though it came from the same platform.

The example shown shows two transport media for GOES-Self-Timed on channel 159 and GOES-Random on channel 129.

You can add or delete transport media by clicking the buttons to the right of the list. Clicking Edit brings up the dialog shown in Figure 7-4.

Note that in this dialog, you associate each transport medium with the name of a "Script" which will be used to decode the data. Scripts are discussed more in section 7.4.

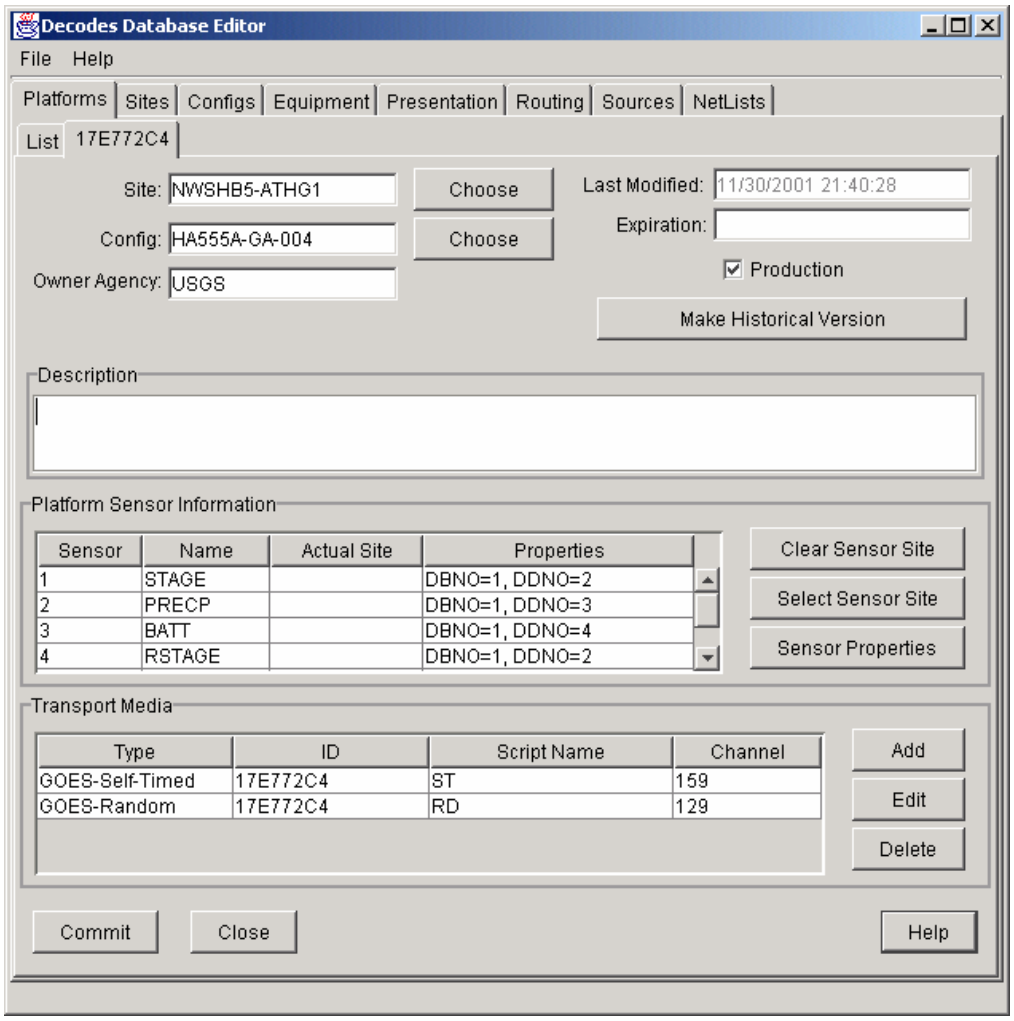

Figure 7-3: Platform Edit Panel.

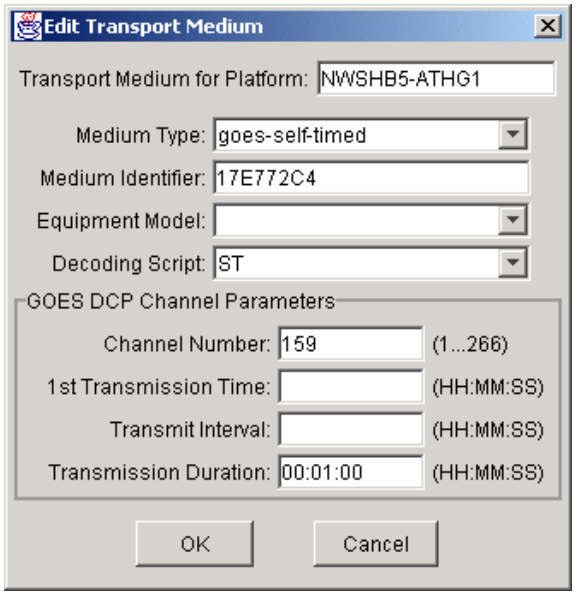

Figure 7-4: Transport Medium Edit Dialog.

7.3 The Site Edit Panel

Figure 7-5 shows an example of the Site Edit Panel. Recall that in DECODES, a site is simply a location with one or more names. The example shows a site near Athens, GA that has three names: A USGS station number of 02217500, a NWS Handbook 5 name of ATHG1, and a DRGS name of "ATHENS". On the right are type-in fields for descriptive information about the site.

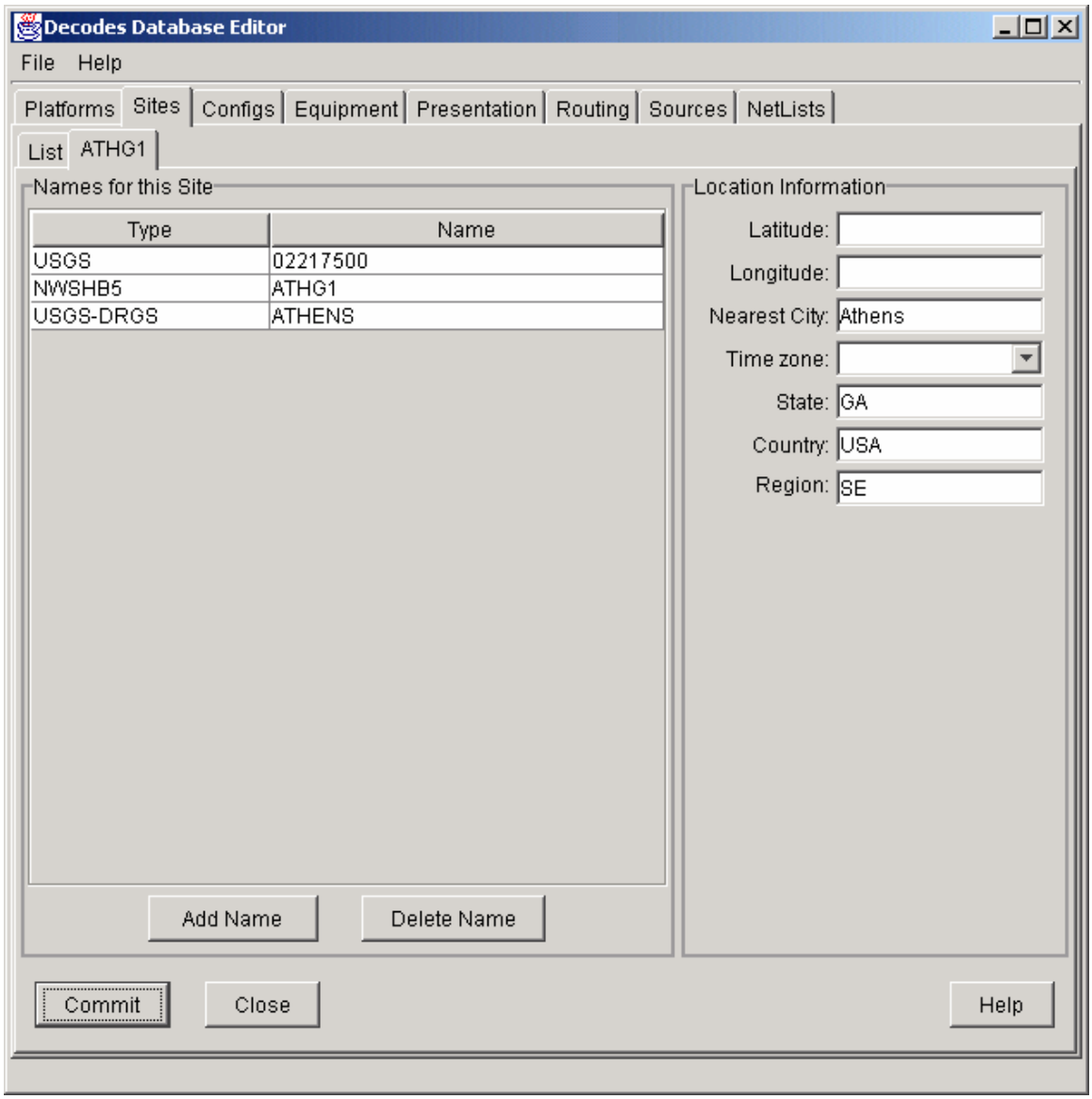

Figure 7-5: The Site Edit Panel.

7.4 The Platform-Config Edit Panel

Figure 7-6 shows an example platform configuration edit panel for a DCP maintained by the USGS.

Configurations are associated with hardware. In this case it is a Handar model 555A DCP. Press the Equipment Model Select button to change this association.

As a convenience, this panel shows you the current number of platforms that are using this configuration. This may be important if you plan to make modifications. Your modifications will effect all platforms using the config.

The center of the panel contains a list of Sensors defined in this configuration. Using the buttons to the right, you can Delete, Edit, or Add sensors in this list. If you edit or add a sensor, you will see the dialog shown in Figure 7-7.

At the bottom of the panel you see a list of decoding scripts. Decoding scripts do the work of extracting sensor samples from your raw message. Using the buttons to the right, you can Delete, Edit, or Add scripts in this list. If you edit or add a script, you will see the dialog shown in Figure 7-8

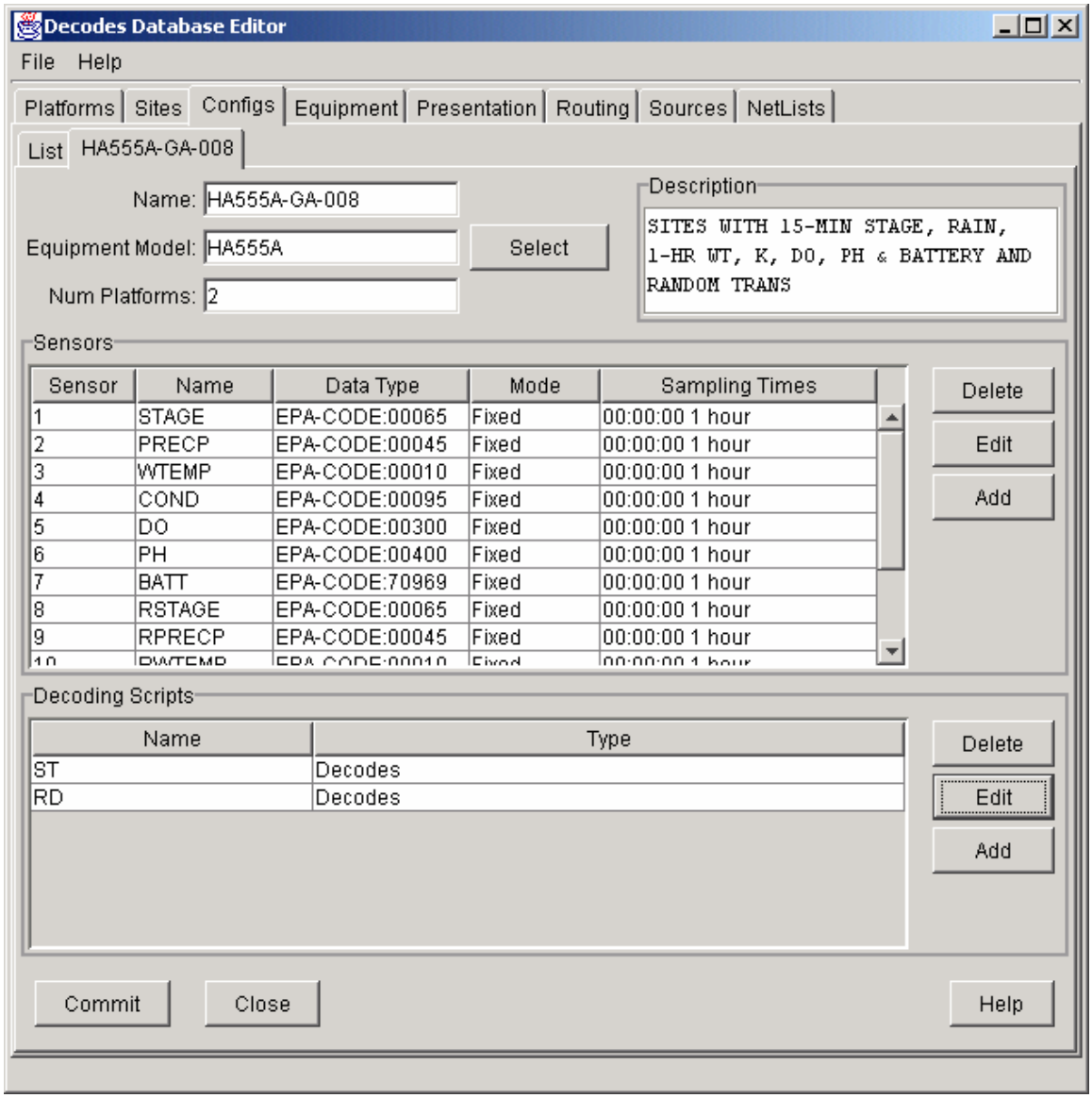

Figure 7-6: Platform Config Edit Panel.

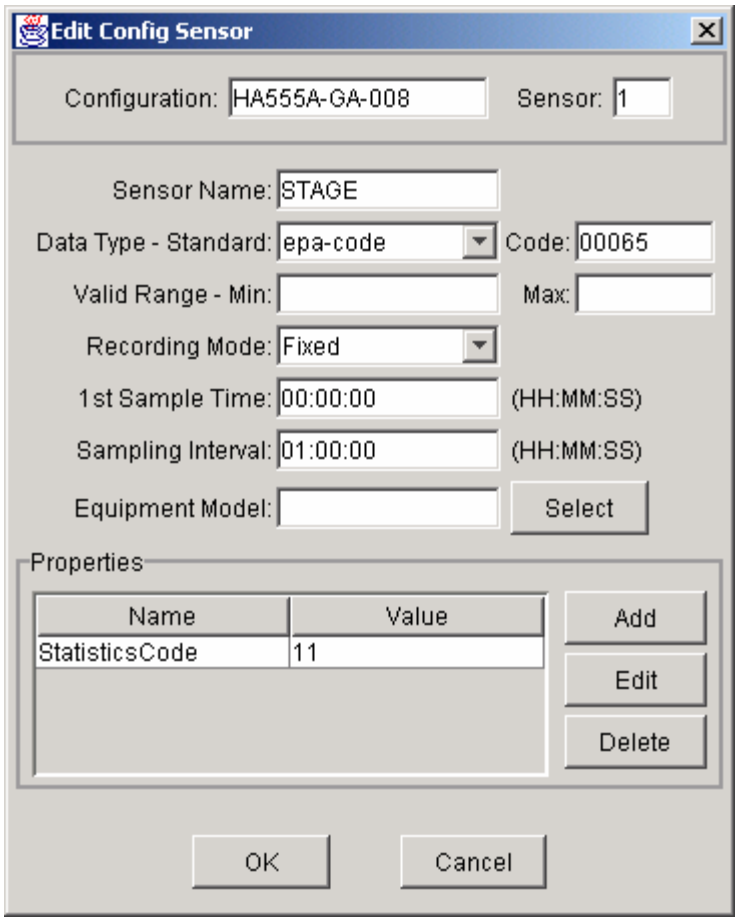

Figure 7-7: Edit Config Sensor Dialog.

7.4.1 The Decoding Script Edit Dialog

This dialog, shown in Figure 7-8 is one of the most important screens in the editor. Anyone who has worked with EMIT or older versions of DECODES will tell you that the hard part is getting the scripts right.

At the top of the screen you see a list of Format Statements, each with a unique label.

The Order of Format Statements is Important! The script will always start with the first statement in the list. You can select a statement and press the Up or Down buttons to move statements around in the list. You can use the Add button to add a new statement at the end of the list. The Delete button will ask you for confirmation before deleting the selected statement.

In the next area of the screen you see a list of sensors. In this list you assign units to each sensor and a raw conversion algorithm. In the example shown the user has selected the algorithm for Battery voltage. Linear conversion (y = Ax + B) has been selected for both parameters. You then type the coefficients directly in the table.

In the Sample Message area you can load raw data and interactively try to decode it using your format statements and conversions. The Load button prompts you for a file containing the raw data. The Clear button clears the text area. You can also manually type into the text area.

If you have LRGS Client Software loaded, there is an even easier way to load sample data. Bring up the LRGS Message Browser and using the search criteria to find and display a message from the desired platform. Now simply select, copy, & paste the data into the Sample Message area of the dialog.

Press the 'Decode' button to apply the format statements to the raw data. The results are shown in the Decoded Data window.

The syntax of format statements is described in section 8, *The DECODES Format Language*.

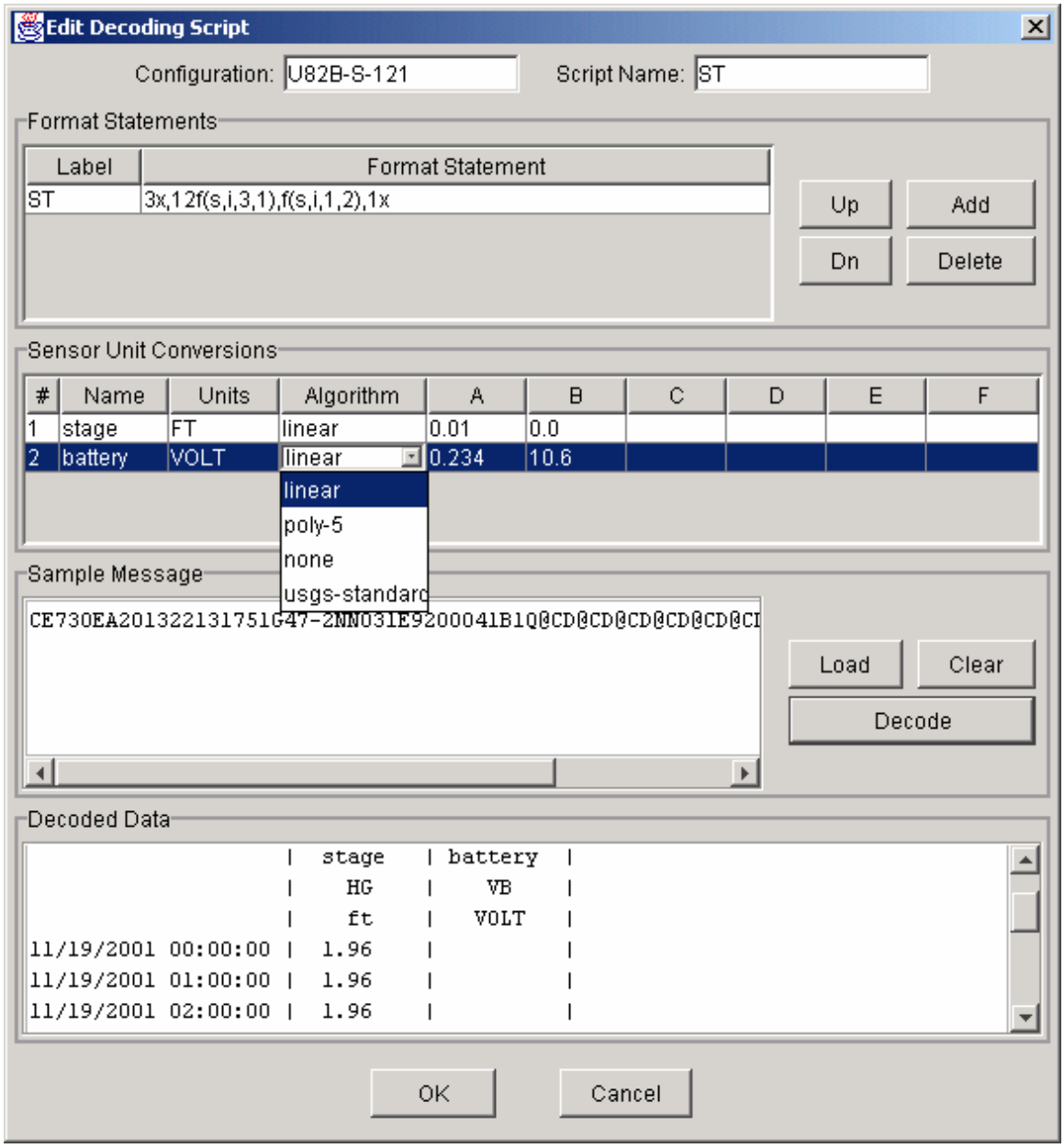

Figure 7-8: Decoding Script Edit Dialog Showing Interactive Decoding.

7.5 The Equipment-Model Edit Panel

Figure 7-9 shows and example of this dialog. Most of the information here is descriptive in nature and not used by downstream DECODES modules.

An exception to this is the "DataOrder" property. If you place this property into an Equipment Model record with the value D (for Descending) or A (for Ascending), then the decoder will apply this to data from platforms using a configuration assigned to this equipment model.

Again the association goes like this: Platform → Config → Equipment Model

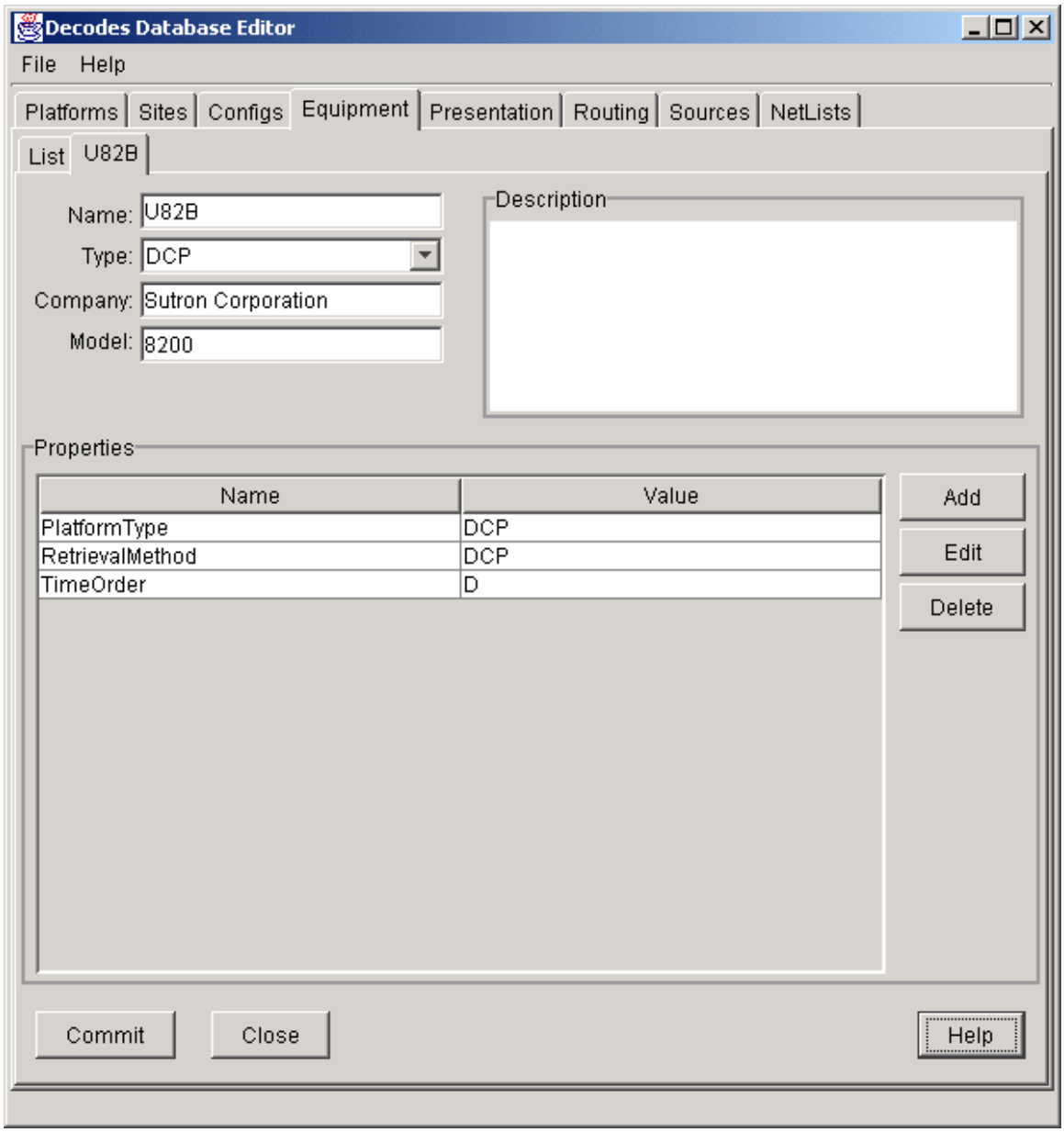

Figure 7-9: Equipment Model Edit Dialog.

7.6 The Presentation Group Edit Panel

An example of this panel is shown in Figure 7-10.

A Presentation Group determines how data will be formatted for output. This includes:

- What engineering units will be used on output.

Numeric rounding rules to apply to each sample value

Look at the example. The first line in the Presentation Elements table says to display HG (stage) values in units of 'ft', or feet. The second line says that PC (precipitation) values are to be displayed in inches.

If you leave the Units field blank, then the value will be output in whatever values are decoded by the script. In other words, no conversion on output will be done.

The third line has a qualifier: It says that HR (reservoir height) values recorded on a Campbell Scientific CR-10B recorder should be displayed in inches. Hence you can get very fined-grained control over display settings.

Finally, notice the last line in this table is for data type "SHEF-PE:*". The '*' means any data type that is not explicitly listed elsewhere.

Notice that in the example the first line of the Presentation Elements table is selected (i.e. highlighted). When you select a presentation element, the Rounding Rules table at the bottom shows the rules to apply to those parameters.

For each rounding rule you specify a maximum value, significant digits, and maximum number of (fractional) decimal digits. Hence, the resolution can change over the possible ranges of values.

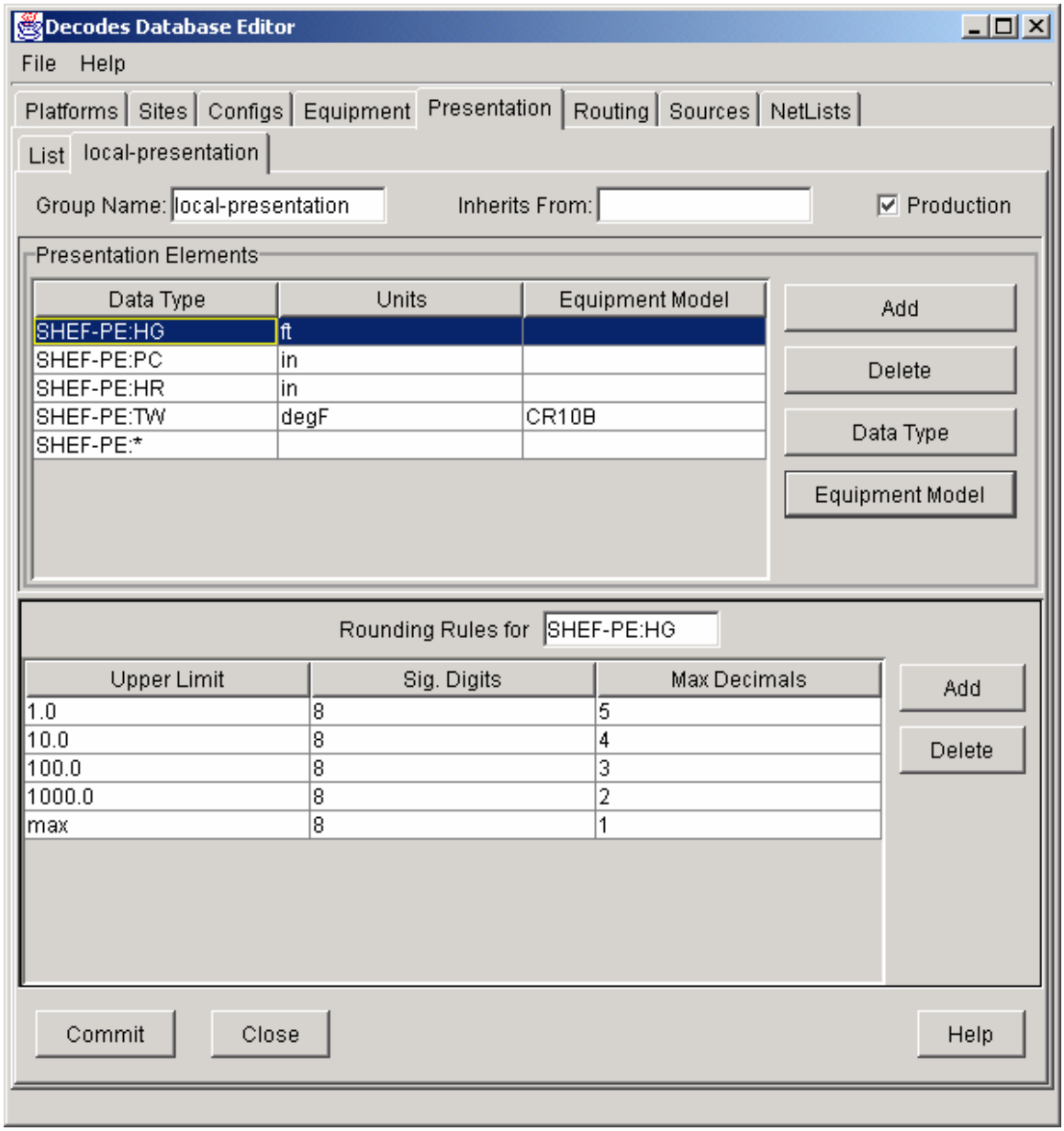

Figure 7-10: Presentation Group Edit Panel.

7.7 The Data Source Edit Panel

Figure 7-11 shows a data source that pulls data from the DROT machine operated by NESDIS at Wallops, VA. Note the properties that are appropriate for LRGS data sources:

- host: host name or IP address of the LRGS or DRS
- username

port

Note that in this figure the Group Members list is disabled. LRGS data sources are not groups.

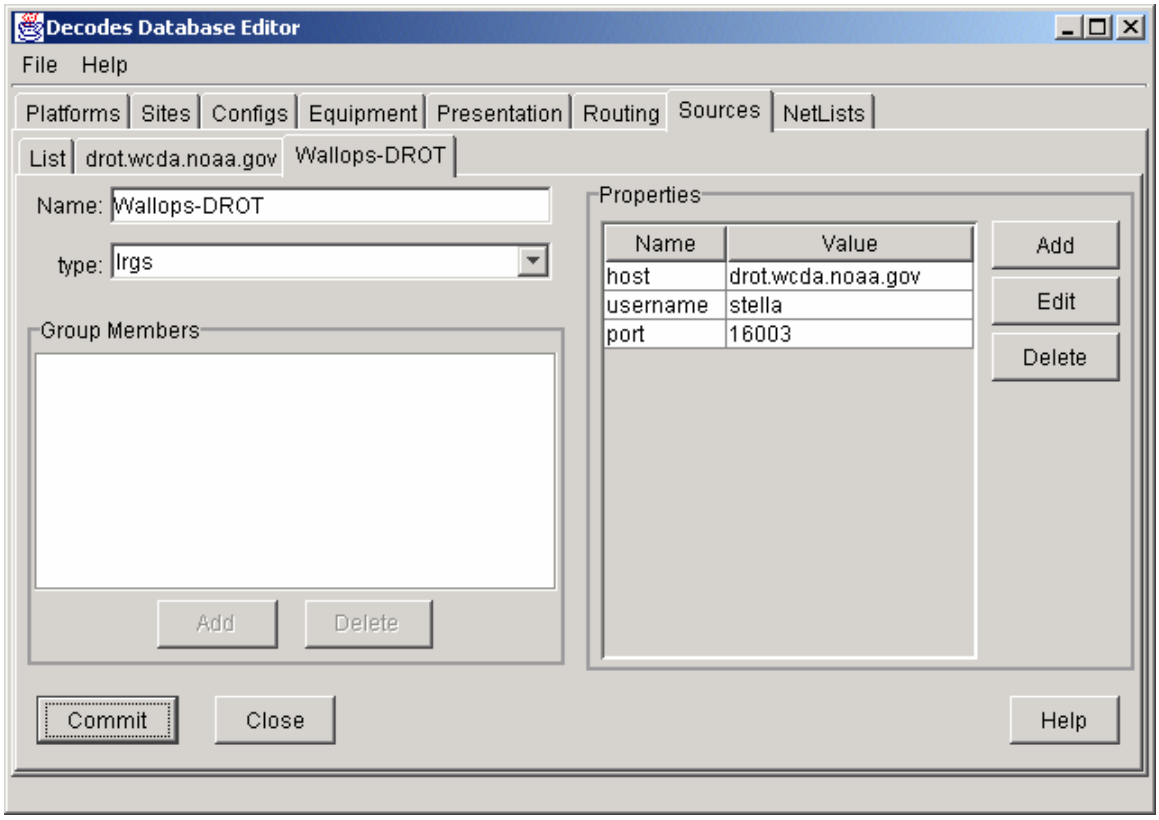

Figure 7-11: Data Source Edit Panel showing LRGS Data Source.

Figure 7-12 shows an edit panel for a hot-backup group. In the example shown, the source will try to pull data first from Wallops-DROT. If unsuccessful, or if it fails in mid-stream, it will automatically switch to another member of the group.

When making a connection, group members are always tried in the order they are specified in the list.

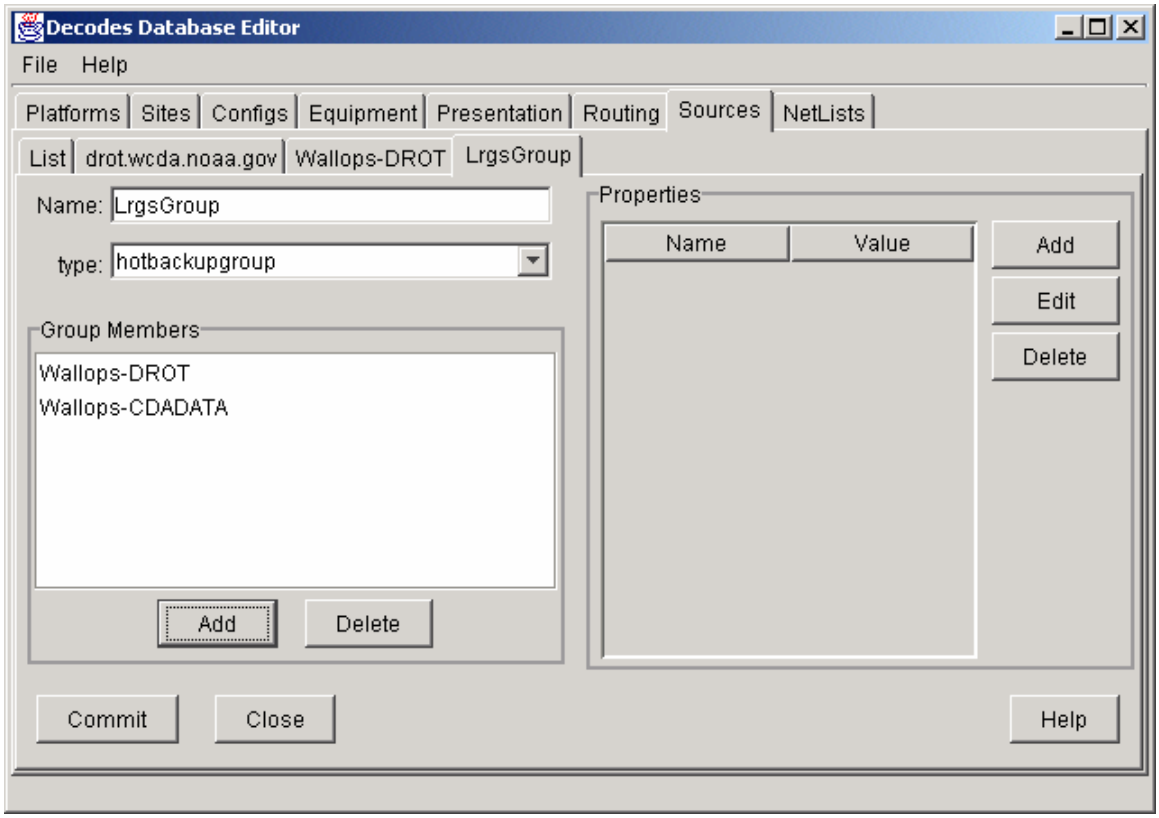

Figure 7-12: Data Source Edit Panel showing Hot Backup Group.

7.8 The Network List Edit Panel

Figure 7-13 shows the StPaul Network List being edited.

A network list is a collection of identifiers for a particular transport medium type. If the transport medium type is "GOES", then the TransportID is a DCP address (as shown). Currently this is the only type of network list in use.

You can add or remove sites from the list using the buttons to the right of the list.

You can click in the headers of the list to cause the list to be sorted by Transport ID, Site Name, or Description.

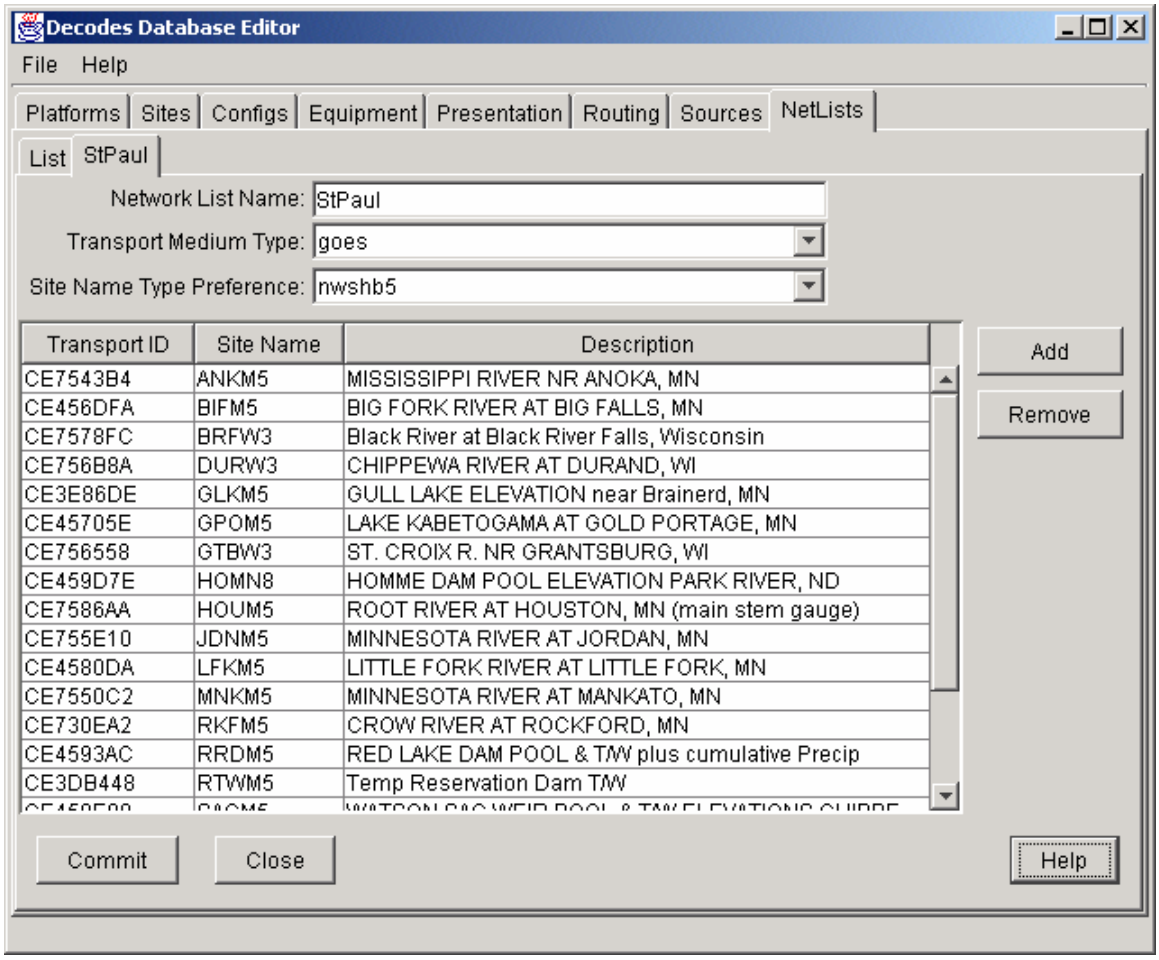

Figure 7-13: Network List Edit Panel.

7.9 The Routing Specification Edit Panel

We saved Routing Specification until last because they tie together all of the other entity types. Figure 7-14 shows a sample routing specification being edited.

The semantics of each field are covered at length in section 8. For now the example shows the following:

- A Routing Spec called “Atlanta-lrgs-input”
- It will read data from the data source called “LrgsGroup” that we saw in Figure 7-12.
- It will send data (consumer) to a pipe to the standard output (stdout).
- It will format data compatible with the EMIT ASCII output format.
- Sample times will be converted to EST
- Data will be presented according to the “local-presentation” group that we saw in Figure 7-10.
- Every time the routing-spec is run, data will be pulled from a time range of “now – 1 day” until “now”.
- There is a property defined called “OldChannelRanges” set to true. This causes the old rule to be in effect that GOES channels less than 100 are self-timed and over 100 are random.
- Only data from platforms referenced in the “Atlanta” Network List will be processed.

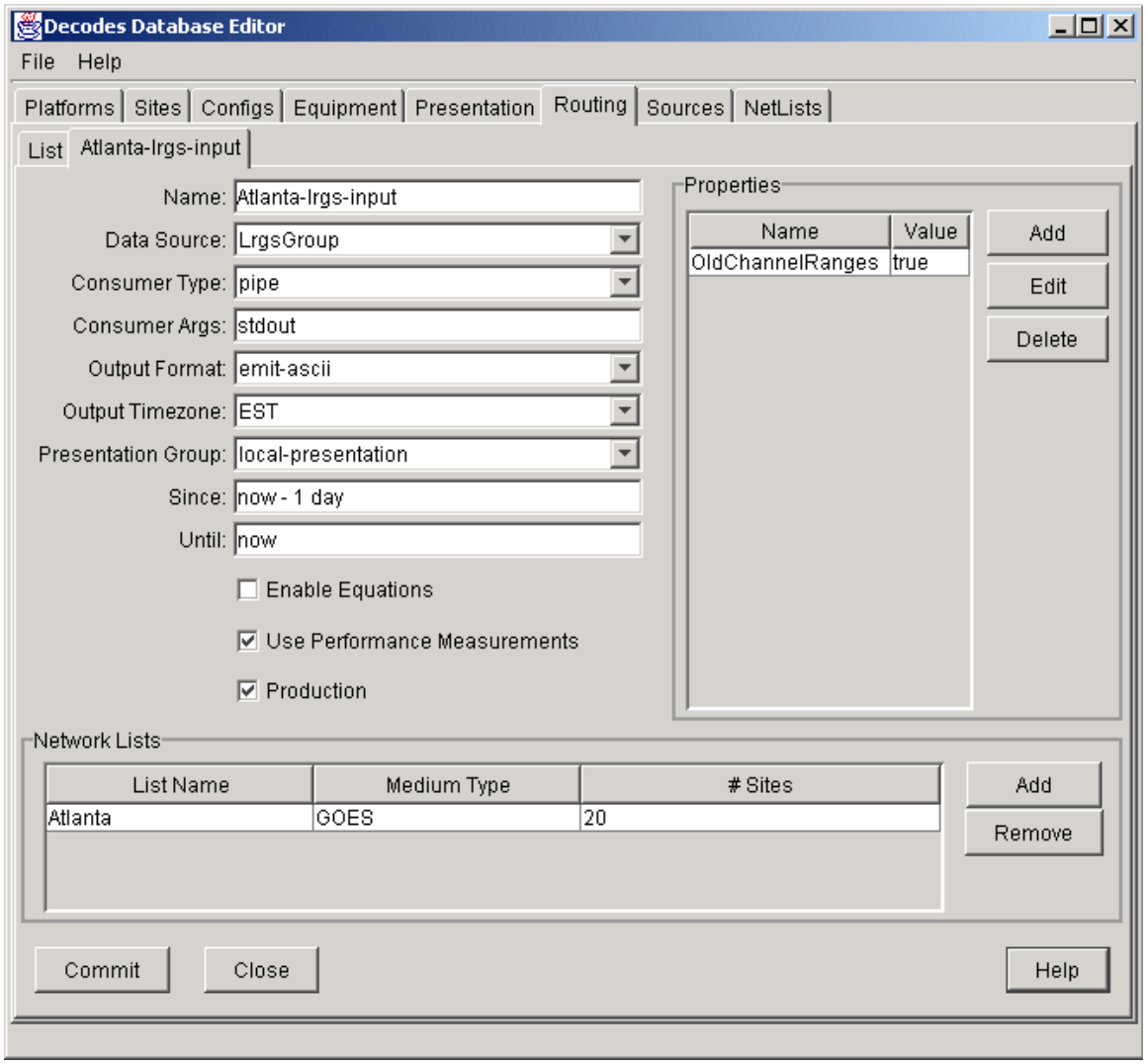

Figure 7-14: Routing Specification Edit Panel.

8. The DECODES Format Language

DECODES uses Fortran-like format statements to interpret data received from a recording device. A *Decoding Script* is made up of one or more format statements. These format statements consist of two parts:

1. a *label* to identify the format, and
2. a *statement* containing a sequence of format operations.

Within a statement, the format operations are separated from each other by commas. You enter format statements within the Decoding Script Edit Dialog, described in section 7.4.1.

8.1 Execution of Format Statements by a Routing Spec

This is what happens when a routing spec decodes a message:

1. Use the DCP Address and channel number within the message to find a matching transport medium.
2. Get the platform record associated with that transport medium. The platform record is associated with a platform-config record, which in turn contains sensor records and one or more Decoding Scripts.
3. Retrieve the Decoding Script associated with this transport medium. For example, the message came in on channel 34, so use the 'ST' (self-timed) script.
4. Parse the script into a hierarchy of executable operations.
5. Starting with the first format statement in the script, execute the operations against the message data.

Step 4 (parsing the script) is only done once. If a second message is received for the same platform, the already-prepared scripts are reused.

Step 5 (executing the script against the message data) is the subject of this chapter.

8.2 Stepping through the Script and the Data

As it is executing, the script keeps track of three things:

- The currently executing format statement
- The current operation within the format statement

The current position within the message data

The message header (e.g. 37-byte DOMSAT header) is not processed by the script. The data pointer is initialized to the first actual message byte.

The script starts with the first format statement, so position is important. This differs from previous versions of DECODES and EMIT.

Each format statement has a label. Several operations can cause decoding to jump to a new statement, identified by its label. Labels may only contain letters and digits.

Older versions of DECODES and EMIT had fixed rules about the labels for self-timed and random messages. Self-timed formats started with the label 'ST', and random message formats started with the label 'RD'. This is no longer required, but it is a useful convention to continue.

Adjacent format statements with the *exact* same label are joined into a single long statement before parsing and executing.

The various operations in the format statements step through the message data from beginning to end. There are operations for skipping characters and lines, and for positioning the data pointer within the message data.

8.3 Format Operations

A quick reference of DECODES format operations is presented in Table 8-1. The subsections that follow provide more detail on each one.

Several of the operators are identified by a letter. The parser is not case-sensitive, so 'X' and 'x' can both be used for skipping characters.

Format Command	***Description***	***Examples***
*n*X	Skip n data characters	2X - *skip 2 characters (bytes).*
*n*P	Position to the nth character in the current line.	2P - Position to 2nd character in current line.
n/	Skip n data lines	3/ - skip 3 lines
n\	Skip backward *n* data lines.	
>*label*	Jump to the format with the specified label	>ST3 -switch to format with label ST3
n(*operations...*)	Repeat operations enclosed in parenthesis n times	10(3X, F(S, A, 6,1)) – repeate "3x,F(s,A,6,1)" 10 times.
C(*n*N, *label*)	Check the next 'n' characters for digits. If all are digits, continue to the next format operation. If at least one is not a digit, switch to format with specified label. Do not change the current data pointer.	C(3N, ERROR) - checks the next three characters for digits. If at least one of the three is not a digit, switch to format ERROR
C(S, *label*)	Check the next character for a sign ('+' or '-'). If it is a sign, continue to the next operation within this format statement; otherwise, switch to the format with specified label. Do not change the current data pointer.	C(S, ERROR) - checks the next character for a sign, switch to format ERROR
C('*str*', *label*)	Compare the string of characters 'str' with the next length-of-string characters in the device data. If there is a match, continue to the next operation in the current format. Otherwise, switch to the format with the specified label. Do not change the current data pointer.	C('001',NXT) - checks the next three characters for a match with '001'. If there is no match, change to format labeled NXT.
S(*n*, N, *label*) S(*n*, S, *label*) S(*n*, A, *label*) S(*n*, '*str*', *label*)	The second argument defines what to scan for: N = scan for any number character (digits) S = scan for any sign character ('+' or '-') A = scan for any alphabetic character '*str*' = Scan for specified string Starting at the current byte, scan at most ***n*** data bytes until either the target of the scan is found or an end-of-line (LF) is found. If the target of the scan is found, continue with the next operation in the current format. Otherwise switch to the format statement with the specified label. After the operation is completed the current data pointer points to where the scan halted, i.e. if target character(s) is found, it points to that character. Otherwise, it is moved 'n' characters form the previous position. A special case of the S operation results when n is 0. In this case the current data pointer remains unchanged. If the target of the scan is found, continue with the next operation. Otherwise switch to specified format. This feature allows multiple tests on the same data character.	S(6,N,ERROR) - scan at most the next 6 characters searching for a number or a sign; and if found, set the data pointer to the matching character and continue to the next format operation; if not found, set the data pointer plus 6 and change to the format with the label ERROR S(0,'A',NXT) - check the current data character to see if it matches 'A'; if it does, continue to the next format operation; if not found, change to format with format label NXT; in either case the data pointer is not changed. S(10,'01+',ERROR)- scan the next 10 characters for the string '01+'. If not found, change to format with label ERROR.
*n*F(*FT, DT, L, S, E*)	Field Descriptions.	Many varieties.

Table 8-1: Format Operations at a Glance.

8.3.1 Skipping and Positioning Operations

To skip a single character:

```
x
```

To skip a specified number of characters, place a number before the 'X':

```
5x
```

To skip to the end of the current line and continue processing data at the beginning of the next line, use a forward slash:

```
/
```

To skip to the end of more than one line, place a number before the slash:

```
2/
```

To position the data-pointer to a particular character position on the line, put a number followed by the letter 'p'. The following positions the pointer to the 5th character of the line. Note: byte position 1 is the start of the line.

```
5p
```

To skip backward a number of lines, use a backslash preceded by a number.

```
2\
```

8.3.2 The Check Operation

Check commands are used to check the current location in the data for a specified condition. If the condition is true, the data pointer is not altered. If the condition is false, you specify an alternate format statement to jump to.

To check to make sure the next *n* characters are numbers (digits), and jump to the statement labeled 'NAN' if any are not, do the following. Note that if the check is true, we proceed with the next operation, which assigns the numbers to a sensor value.

```
c(5N, NAN), f(s,a,5,1)
```

To check if the next character is a sign (either '+' or '-'), and jump to the statement NOSIGN if not:

```
c(S, NOSIGN), ...
```

To check to see if the data matches the string 'AA' and skip to the format labeled 'BB' if it does not:

```
c('AA', BB), ...
```

In this usage of the check command, the string must match exactly. The check is case sensitive and the entire string must match the current data position. Otherwise the check is false and control jumps to the named format statement.

8.3.3 The Scan Operation

Scan commands are used to scan forward from the current location in the data until a specified condition has occurred. These commands are used to position to a particular location based upon a specified condition.

Scan operations have the following syntax:

```
S(n, condition, label)
```

…where ***n*** is the number of characters to scan (or to the end of the current line), ***condition*** specifies what we are scanning for (see below), and ***label*** specifies the format that we jump to if the condition is not met.

The ***condition*** can be one of the following:

`N`	Scan for any digit
`S`	Scan for any letter, either upper or lower case
`Xnn`	Scan for a character with the hex value *nn*
`'str'`	Scan for the exact string ‘str’

If the condition is true (i.e. the requested pattern was found), processing continues to the next operation in the current format statement. The data pointer is left at the first character that matched the scan. For strings, the data pointer is left at the first character of the string.

8.3.4 The Jump Operation

The Jump operation causes an unconditional jump to a specified format statement. The data pointer remains unaffected. The jump operation has the following syntax:

```
>label
```

8.3.5 The Repeat Operation

Any group of operations can be performed repeatedly. Operations enclosed in parentheses and preceded by a number will be performed the specified number of times. For example,

```
8(x,F(S,B,3,1))
```

causes the operations within the parentheses (the **x** operation and the **F** operation) to be performed 8 times.

8.3.6 Field Operations

Field operations are used to extract time and sensor values from the message. The general form of a field description is:

***n*F(*ft*, *dt*, *len*gth ,*sensor # or fld-ID*, *E*)**

where:

- ***n*** is a repetition factor
- ***ft*** defines the type of field
- ***dt*** defines the type of data
- ***length*** defines the field length with optional delimiter.
- ***sensor #*** defines the sensor number associated with this field (only used in sensor-value fields)
- ***fld-id*** is used with DATE and TIME fields to specify different representations

E is used with TIME fields to indicate that the recording of time should be viewed as an event

The field type can be one of the following:

D	Date Field (see 2.2.4.8.1)
T	Time Field (see 2.2.4.8.2)
TI	Time Interval Field (see 2.2.4.8.3)
F	Format Label Field (see 2.2.4.8.4)
S	Sensor Value Field (see 2.2.4.8.5)

The data type can be one of the following:

A	ASCII
B	Binary (unsigned)
I	Integer (signed binary)
L	Labarge pseudo-ASCII
X	Hexadecimal
S	String
BC	Campbell Scientific Binary Format
C	Campbell Scientific Binary Format (first byte defines sign and magnitude)
BD	Design Analysis binary Format (Integer value made negative by sign bit)
BT	Telonics Binary Format (Integer value made negative by sign bit)

Field Length and Delimiters:

Length can be optionally followed by the character D and a delimiter character. For example:

```
6D,
```

This indicates that the field has a length of 6 characters or can be delimited by a comma.

The delimiter can be simply asserted (as in the example above), enclosed in single quotes, or represented as *xnn* where nn is the hexadecimal representation.

For example:

6D,	The field is delimited by a comma
6Dx1E	The field is delimited by a period (the hexadecimal representation of a period is 1E).
6D' '	The field is delimited by a space

If the character after the 'D' is 'S', it means that the data is delimited by a sign (+ or -).

Care must be taken in positioning your data pointer after a delimited field. The pointer will be left *at* the delimiter (unless you place an 'X' after the delimiter, see below). Hence you will probably want to use a skip operation to skip the delimiter after parsing the field.

If the delimiter is not found, the pointer is advanced by *length* characters.

Placing an 'X' character is used to indicate that the delimiter should be automatically skipped obviating the need for a format command of 1X after each field description.

8.3.6.1 Date Fields

Date field descriptions have a field type of 'D'. Date fields are used in EDL files to extract time from the message data. The times are then subsequently used to time-tag data samples.

The form of a date field description is

F(D, data type, length<Dc>, fld-id)

The 'fld id' parameter is used to define four different date formats. Possible formats are as follows:

F(D, type, length<Dc>,1)

Fld-id 1 indicates the date is basically in the format year, month, day. The format differs slightly for different field lengths. For length 8, fields have the format YY/MM/DD, YY-MM-DD, and YY MM DD; for length 6, fields have the format YYMMDD.

F(D, type, length<Dc>,2)

Fld-id 2 indicates a Julian day is used. For length 8, fields have the format YYYY-DDD, YYYY/DDD; for length 7, YYYYDDD; for length 6, YY-DDD, YY/DDD; for length 5, YYDDD; for length 3, DDD; for length 2, DD. For cases where the year is not in the date field, the year will default to the current year unless the user specifies a year during the data conversion process. If the user lets the year default and a Julian day is found that

exceeds the current Julian day, it will be assumed that the data belongs to the previous year and so the year will be decremented.)

F(D, type, length<Dc>,3)

Fld-id 3 indicates only the month and day are recorded. For length 5, fields with format MM/DD, MM-DD, AND MM DD; for length 4, MMDD. The same rules about the missing year apply to the field descriptions for dates with fld id of 3 as the ones for the dates with fld id of 2.

F(D, type, length<Dc>,4)

Fld-id 4 indicates the same type of format as fld-id 1 but in a different order-month, day, year. For length 8, fields with format MM/DD/YY, MM-DD-YY, and MM DD YY; for length 6 MMDDYY.

You can also parse the date components individually:

`F(YR, type, length)`	Parse a year field. Length can be 2 or 4.
`F(MN, type, length)`	Parse a month field. If length is 2, expect a number from 1 to 12. If length is 3, expect a 3-character month abbreviation like jan, feb, etc.
`F(DY, type, length)`	Parse day of month.
`F(JDY, type, length)`	Parse julian day-of-year.

8.3.6.2 Time Fields

Field descriptions for times have a field type of 'T' and a data type of 'A' (ASCII). Thus, the form of a field description for a time is

F(T, A, length<Dc><, sensor #, E>)

The optional *'sensor #'* and *'E'* parameters signify that the time recorded is an event. This is used for recorders that record only the time whenever an event occurs e.g. the time is recorded whenever a tipping bucket tips. In this case, the recorded time is considered to be the data. When DECODES encounters a field description for a time and it has a sensor number and the 'E' parameter, DECODES will use the value 1 as the data value associated with that time.

The raw value of 1 can be converted to the desired units via an EU conversion in the script. For example, if a tipping bucket rain gage records the time whenever .01 inches of rain falls, convert the raw value of 1 to .01 with a linear EU conversion.

For length 8, times are expected with format HH-MM-SS or HH:MM:SS; for length 6, HHMMSS; for length 5, HH:MM, HH-MM; for length 4, HHMM.

8.3.6.3 Time Interval Fields

Time interval field descriptions have a field type of TI and a data type of 'A' (ASCII). The time interval field describes a field that contains a new time interval for recording data. This field description is useful for recorders that can adjust the recording interval from that set in the SENSORS entity to a new one when certain conditions occur. The form of a field description for a time interval is

F(TI, A, length<Dc>)

The format of the data field that it describes is the same as those for the time field description.

8.3.6.4 Format Label Fields

Format-label fields describe a data field that contains a code that is to be used as a format label to select a new format. DECODES extracts a label from the message data and jumps to a matching format statement.

The data pointer will remain at the character immediately following the extracted format-label.

Format-label fields allow DECODES to switch formats based upon a code found in the device data. For example, if a device records the data in different formats and also records a code that identifies the each format, a statement can be written for each code, using the code itself as a format label.

If DECODES cannot find a match for the label extracted from the data, it will attempt to switch to a format statement with the label 'ERROR'. If none exists, decoding of this message will be aborted.

The format of a field description for format labels is

F(F, A, length<Dc>)

Examples:

F(F, A, 4) - Format label field is 4 characters long.

F(F, A, 8D,) - Format label field is delimited by a comma and has at most 8 characters.

8.3.6.5 Sensor Value Fields

Sensor field descriptions have a field type of 'S'. They are used to extract data samples from the message. The format of a sensor field description is

nF(S, data type, length<Dc>, sensor #)

"Data type" can be one of the following:

- A ASCII
- B Binary (unsigned)
- I Integer (binary signed)
- X Hexadecimal

Examples:

F(S, A, 6, 1)	The Data will contain one 6-character ASCII sample for sensor number 1.
F(S, A, 5D,,2)	The data is delimited by a comma and has at most 4 ASCII characters; the value was produced by sensor 2.
3F(S, B, 3, 1)	3 signed-binary samples for sensor number 1. Each sample is 3 characters long.

9. DECODES Routing Specifications

Figure 9-1 shows the data flow for a routing specification. Take a moment to study the components involved. This section will discuss how to run a routing specification and how to control each of the components shown in the figure.

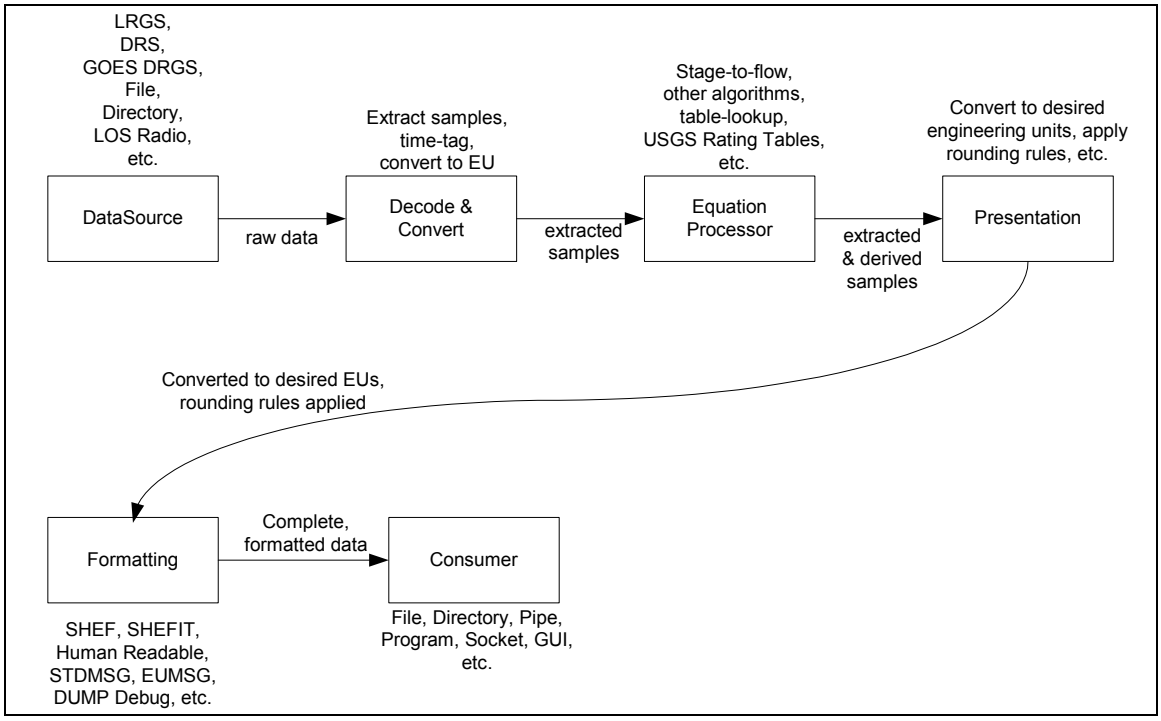

Figure 9-1: Data Flow for Routing Specifications.

9.1 How to Run a Routing Specification

Synopsis:

```
rs <options> spec-name
```

Options:

`-e`	Run from the editable database (default is installed database)
`-s`	Script name to be executed. This option may appear multiple times. This can also be accomplished with the "scriptname" property.
`-m`	Do NOT apply sensor min/max limits (default is to do so).

Description:

This script starts a Java Virtual Machine running the specified routing spec. All of the parameters that control the action of the routing spec are specified in the database or the DECODES properties file. Hence there are no options to this command.

Routing Spec Properties can be used to control the execution of the spec, or to control the actions of various component objects. The properties which apply at the top level are:

- scriptname: This is a blank-separated list of script names to be executed. The default action is to execute any script. This property is equivalent to the -s command line argument.

nolimits: Value is either true or false (default = false). This property is equivalent to the -m command line argument. It tells the spec to NOT apply sensor min/max limits, even if they are defined.

Examples:

`rs Atlanta-lrgs-input`	*Execute routing spec "Atlanta-lrgs-input" from the installed database.*
`rs -e test`	*Execute routing spec "test" from the editable database.*
`rs -e -s ST test`	*Execute routing spec "test" from the editable database, but only process messages for ST (self-timed) scripts.*

9.1.1 Routing Spec Properties

In the database editor you can enter properties that affect various aspects of a routing spec. Currently, none of these properties are used by the routing spec itself. Rather they are passed to the components of the routing spec like Data Source, Consumer, Output Formatter, etc. The properties used by components are described in the sub-sections that follow.

9.2 Data Sources

The following sections describe the semantics of data sources. See section 7.7 for instructions on how to modify a data source's parameters.

9.2.1 LRGS Data Source

LRGS Data Sources are used to connect to LRGS or DRS systems over the network. The LDDS Server must be running on the LRGS you want to connect to.

Properties for the LRGS Data Source may be placed in the Data Source record or the Routing Spec Record in your DECODES database. Properties defined in the Routing Spec record will override those of the same name defined in the Data Source record.

So, for example, if the Data Source record contains "username=joe", but the Routing Spec record contains "username=ted", THEN "ted" will be the username passed to the LRGS server.

Accepted properties are as follows:

- host: The host name or IP Address of the LRGS system to connect to. (Optional, If missing, the name of the data source object is used.)
- port: Port number for this LRGS's server. (Optional, default = 16003)
- username: registered user on the LRGS server (required)
- password: Some LRGS servers are configured to require passwords. If this is the case, you will need to enter the password here. ***Warning! The password will be stored in clear text in the SQL database and XML files.***
- single: (Default=false) The newer LRGS servers have a new feature whereby many DCP messages can be returned for a single request. By default, DECODES will use this feature if the server supports it. To force the old (single message per request) behavior, add a property "single" with a value of either "on", "true", or "yes".
- sendnl: (Default=true) – Old DRS servers do not support network list transfers. Set this to false when connecting to such servers. The data source will then assume that the network lists are already loaded on the DRS. You must then transfer the list using some other mechanism (e.g. FTP) prior to running the routing spec.
- response.timeout: (Default=60 seconds) This is the number of seconds to wait for a response from this server. See discussion of timeouts below.
- searchcrit: If supplied, this should be the full path-name to a search criteria file to be passed to the server. See below on how searchcrit information is processed.

Each time an LRGS is initialized, it is passed the new search criteria from the routing specification. This information includes the "since" and "until" times, network lists, and the routing spec properties.

The Routing Spec may contain a property called "lrgs.timeout", set to a number of seconds. If so, this value will be used by the LRGS data source. The default timeout is 60 seconds.

The routing spec will exit with the LRGS Data Source determines that the specified "until time" has been reached. If no until time is specified, the routing spec will continue running indefinitely.

The "searchcrit" Property

If you supply a "searchcrit" property in either the routing spec or data source record, it will be the full path-name to a search criteria file. This allows you to use the full range of search-criteria in a routing spec.

The information in the file will be somewhat modified before being passed to the server. If the routing spec contains a "since" or "until" time, these will override the values in the search criteria.

If the searchcrit names a network list, with or without the ".nl" extension, then:

- IF the list is contained in your DECODES database, it will be sent to the server.
- ELSE IF the list is found on your hard disk in either the current directory or a subdirectory called "netlist" (i.e. in "." or "./netlist"), then it will be sent to the server.

ELSE, assume that the list already resides on the server.

Also, any network lists specified directly in the routing spec will be sent to the server.

9.2.1.1 Timeouts in LRGS Data Sources

There are two timeout values that effect the operation of an LRGS Data Source:

The "response.timeout" property in the LRGS Data Source object controls how long to wait for a response from the server after sending a request. The purpose of this timeout is to catch connections that have failed. For example, the server is no longer responding or a WAN link has gone done.

The "lrgs.timeout" property *in the Routing Spec object*, specifies the maximum number of seconds to wait for the next message to arrive. This means, even if a link is up and the server is responding to each request in a timely fashion, wait no more than this many seconds for the next message. The purpose of this timeout is to catch problems upstream from the server.

The "lrgs.timeout" property is associated with the routing spec (not the Data Source) because it depends on what data you are retrieving. For example, if I am getting data from a single DCP that reports hourly, I might set lrgs.timeout to 3660 (1 hour and 1 minute).

In most cases, the "response.timeout" should be fairly low. The default value of 60 seconds should suffice.

When a timeout (of either type) occurs, the LRGS Data Source throws an exception and…

- If this LRGS is part of a Hot Backup Group, the group will attempt to connect to another LRGS.
- If this LRGS is the sole data source, the routing spec will terminate.

9.2.2 File Data Source

A File Data Source reads a series of DCP messages from a single file. It processes the file from beginning to end and returns each message found therein. After reaching the end of the file, the Data Source causes the routing spec to exit.

Accepted properties for a File Data Source are as follows:

- filename: If present, this value will be used as the file name to be read. It can be a complete path name or a filename relative to the current working directory. If this property is absent, the name of the data source will be assumed to be a file name.
- before: A special string that delimits the beginning of a new message in the file. This string may contain binary and escaped characters such as \n (newline) or \001 (ASCII STX).
- after: A special string the delimits the end of a message in the file.
- mediumType: Specifies the type of data stored in the file. (Optional, default is "GOES").
- mediumId: Specifies the transport medium ID of the platform that generated the messages in the file. Optional: Only use this if all the messages in the file came from the same platform. This is primarily used for EDL files collected in the field.

9.2.2.1 Delimiting Messages Within the File

The 'before' and 'after' strings are optional. Here is how DECODES interprets them:

- If neither 'before' or 'after' is specified, the entire file is assumed to contain a single message.
- If 'before' is specified, but 'after' is not. DECODES will scan the file for the 'before' string and return data following it, up to, but not including the next 'before' string. The final message terminates at end-of-file. Any data in the file prior to the first 'before' string will be ignored.
- If 'after' is specified, but 'before' is not. The first message starts at the beginning of the file and continues up to, but not including, the first occurance of the 'after' string. ny data at the end of the file not terminated by the 'after' string will be ignored.
- If both 'before' and 'after' are specified, only completely delimited messages will be processed from the file.

9.2.3 Directory Data Source

NOT YET IMPLEMENTED!

A directory data source continually scans a specified directory. When files are placed in the directory they are immediately processed with a dynamically generated "File Data Source". For this reason, the before, after, mediumType, and mediumId properties work and have the same meaning as they do for File Data Source.

The files must be complete when they are placed in the specified directory. So, if you are constructing the file by reading data from a socket or serial line, construct a temporary

file in a different directory. When the file is complete, move it to the directory specified for the Directory Data Source.

9.2.4 Hot Backup Group Data Source

A Hot Backup Group Data Source is primarily used for a set of LRGS connections. One connection may fail, in which case we want our routing spec to try another. This makes your routing spec more reliable, particularly if this is a real-time routing spec that runs continuously (i.e. no "Until Time").

Currently there is only one property that is used by a Hot Backup Group:

- recheck: (default = 900 seconds, or 15 minutes) – If the currently active data source is not the first one in the list, the Hot Backup Group will attempt to connect to higher priority data sources at this period.
- fudge: (default = 120 seconds, or 2 minutes) – Amount of time to back-up after connecting to new data source.

The Hot Backup Group contains an *ordered* list of LRGS data sources. The group will prefer the members in the order they are listed.

Upon start-up, the group will attempt to connect to a LRGS, starting with the first one listed. Once a successful connection is made, this LRGS becomes *active*. The group then reads DCP messages from this source until…

- The active source fails (either a timeout or broken connection), or
- The active source is not first in the list *and* the recheck period expires.

When this happens, the group will try to connect to a source, once again starting from the first in the list.

When the group changes from one active source to another, it passes the new source the network lists and search criteria with one modification: The 'since' time is adjusted to:

LastMessageTime – fudge

… where LastMessageTime is the time of the last DCP message I received. The 'fudge' factor (default=120 seconds) can be controlled via a property setting.

The purpose of this fudge factor is to account for small variations in the system clocks of the LRGS members. If you have all your systems synchronized via NTP you can make the fudge factor very small.

Larger fudge factors may result in duplicate messages: A DCP message received from one LRGS and then after a switch, the same message received from the new LRGS.

9.2.5 Round Robin Group Data Source

NOT YET IMPLEMENTED!

A round-robin group contains a list of other data sources.

The purpose of a round-robin group is to continually read data from all data sources in the group. This differs from a hot-backup group, which only uses one data source at a time

9.2.6 Socket Stream Data Source

A socket stream data source opens a socket and reads a one-way stream of data containing raw DCP messages. Some DRGS and DOMSAT product provide such a stream.

Accepted properties for SocketStreamDataSource are:

- host = the host name or IP address of the server
- port = the port number of the socket to be opened
- lengthAdj = a negative or positive number. The default value is -1. (See below)
- delimiter = A string that begins each message, use \r for carriage return and \n for linefeed. The default delimiter is \r\n. (See below)
- endDelimiter = A string that marks the end of each message. This is required if header is "noaaport". The NOAAPORT message format determines the message length not from the header but from the beginning and end delimiters.
- header = GOES, VITEL, NOAAPORT, Vaisala. The default is GOES (See below)

Delimiters and Length Adjustments

Each message must start with a 37-byte DOMSAT header. The last 5 bytes of the header is the number of message bytes to follow. Immediately following the message data, a delimiter is expected. The delimiter is not included in the message length.

The Vitel DRGS reports a message length which is actually 4 more than the number of bytes actually present in the message data. Each message is terminated by a carriage return and linefeed. Hence the proper settings for a Vitel DRGS are:

```
lengthAdj = -4
delimiter = \r\n
```

The DataWise DOMSAT system reports a length that is one greater than the number actually present. It terminates each message with 3 sets of carriage-return/linefeed. The proper settings for a DataWise DOMSAT socket stream are:

```
lengthAdj = 0
delimiter = \r\n\r\n\r\n
```

How messages are parsed

The socket is opened. The input software expects the stream to start with a message header, followed by the message data, followed by the delimiter. This cycle repeats indefinitely until the socket is closed.

The input software can get out of sync in one of the following ways:

- Detecting an invalid 37-byte header (no DCP address, channel number, or message length).

Failing to find the delimiter string

When this happens, the input software goes into "hunt mode". It will read characters from the socket looking for the delimiter sequence. Once found it will again attempt to read the 37 byte header.

Look at the debug-log when running the routing spec. If your 'lengthAdj' and 'delimiter' parameters are correct you will never see the messages saying that the software has skipped data. If you do see these messages:

- Consult the manual for the server system to determine how messages are formatted.
- Make sure the delimiter string is correct as described above.

Try adjustin lengthAdj downward, into negative numbers (incrementally).

Network Lists and Time Ranges

Since a socket-stream is assumed to be a real-time data source, the input software will ignore the 'since' and 'until' times specified in the routing spec.

Network lists will be used to filter incoming data. Only messages whose DCP address is contained in one of the routing-specs network lists will be processed. If the routing spec contains no network lists, all data will be processed.

Header Format

The "header" property should be one of "GOES", "VITEL", or "NOAAPORT". The default is "GOES" if the property is missing. The Vitel header is slightly different in that it does not include the failure-code field, causing subsequent fields to be shifted one character to the left.

9.2.6.1 Using SocketStreamDataSource for NOAAPORT

NOAAPORT messages are received over a socket in the following format:

[SOH]\r\r\n*NNN*\r\r\n*HHH[RS]DDD*\r\r\n*[ETX]*

…where

- *[SOH]* is an ASCII Start-Of-Header character (octal \001)
- *NNN* is a NOAAPORT 3 digit sequence number
- *HHH* is a NOAAPORT Header (ignored)
- *[RS]* is an ASCII Record-Separator character (octal \036)
- *DDD* is the DCP message containing time stamp and other header fields before and after the message proper.
- *[ETX]* is an ASCII End-of-Text character (octal \003)

The *DDD* data field contains all the header fields and message-data that we need. We want to ignore everything else. Consequently use the following Data Source Properties:

- host
- port =
- delimiter = \036
- endDelimiter = \r\r\n\003

header = NOAAPORT

The Socket Stream will then process only the DDD (data) field between the *[RS]* and \r\r\n*[ETX]*, and ignore everything else.

The Data Field itself will have the following format:

```
AAAAAAAA DDDHHMMSS ddd... SSFFNN CCCs
```

…where

- AAAAAAAA is the 8-hex-char DCP Address
- DDDHHMMSS is the date/time stamp.
- ddd… is the actual message data
- SS is the signal strength
- FF is the Frequence offset
- NN is a placeholder for IFPD (it is always set to 'NN')
- CCC is the GOES Channel number, padded on the left with blanks (3 characters)
- s is the GOES Spacecraft (E or W)

9.3 Output Formatters

DECODES supports a variety of output formats including:

- SHEF – Standard Hydrometeorologic Exchange Format .A lines)
- SHEFIT – Intermediate format defined by the USACE Hydrologic Engineering Center. Used to input data into the Corp's CWMS database.
- Human Readable - Simple but compact time-sorted table format
- EMIT-ASCII - Compatible with EMIT when "ASCII" format is selected.
- EMIT-ORACLE - Compatible with EMIT when "ORACLE" format is selected.
- Dump – Used primarily for trouble-shooting, this format dumps all known information about samples, sensors, & platform.
- STDFMT - Standard format used by USGS for data-ingest into NWIS

TransmitMonitor - Displays log of transmission quality parameters and battery voltage.

The following subsections contain more details on individual formats.

9.3.1 SHEF Output Format

The SHEF Output Formatter can produce either the ".A" or ".E" type lines:

- .E is normally used for regular interval data, such as is found in self-timed DCP messages. Figure 9-2 shows an example of the SHEF .A.

.A is normally used for irregular interval data, such as is found in random DCP messages. Figure Figure 9-3 shows an example of SHEF .E.

To force only .A lines to be used in the output stream, add a property to your routing spec with the name "dotAOnly" and a value of "true".

SHEF time stamps allow 4 digit or 2 digit years. The default is a 2 digit year. To force the century to be included, add a property to your routing spec with the name "century" with a value of "true.

Likewise, seconds can be omitted in SHEF time stamps. By default they are included. To force them to be dropped, add a routing-spec property named "seconds" with a value of "false".

```
.A BRFW3 011203 GMT+00:00 DH110000 /DUE /HG  38.36   :ft
.A BRFW3 011203 GMT+00:00 DH100000 /DUE /HG  38.35   :ft
.A BRFW3 011203 GMT+00:00 DH090000 /DUE /HG  38.34   :ft
.A BRFW3 011203 GMT+00:00 DH080000 /DUE /HG  38.35   :ft
.A BRFW3 011203 GMT+00:00 DH070000 /DUE /HG  38.35   :ft
.A BRFW3 011203 GMT+00:00 DH060000 /DUE /HG  38.35   :ft
.A BRFW3 011203 GMT+00:00 DH050000 /DUE /HG  38.35   :ft
.A BRFW3 011203 GMT+00:00 DH040000 /DUE /HG  38.35   :ft
.A BRFW3 011203 GMT+00:00 DH110000 /DUS /PC  6.26    :INCH
.A BRFW3 011203 GMT+00:00 DH100000 /DUS /PC  6.26    :INCH
.A BRFW3 011203 GMT+00:00 DH090000 /DUS /PC  6.26    :INCH
.A BRFW3 011203 GMT+00:00 DH080000 /DUS /PC  6.26    :INCH
.A BRFW3 011203 GMT+00:00 DH070000 /DUS /PC  6.26    :INCH
.A BRFW3 011203 GMT+00:00 DH060000 /DUS /PC  6.26    :INCH
.A BRFW3 011203 GMT+00:00 DH050000 /DUS /PC  6.26    :INCH
.A BRFW3 011203 GMT+00:00 DH040000 /DUS /PC  6.26    :INCH
```

Figure 9-2: Example of SHEF .A Output

```
.E SSIM5 020212 GMT DH150000 /DUS /VB/ DIH+1 /14.344 :V
.E LFKM5 020212 GMT DH080000 /DUE /HG/ DIH+1 /2.79/2.79/2.79/2.79/2.79/2.79/2.79/2.79 :ft
.E LFKM5 020212 GMT DH150000 /DUE /VB/ DIH+1 /14.344 :VOLT
.E VRNN8 020212 GMT DH150000 /DUE /VB/ DIH+1 /13.876 :VOLT
.E BRFW3 020212 GMT DH080000 /DUE /PC/ DIH+1 /6.26/6.26/6.26/6.26/6.26/6.26/6.26/6.26 :in
.E BRFW3 020212 GMT DH150000 /DUS /VB/ DIH+1 /14.5 :V
.E DURW3 020212 GMT DH080000 /DUE /HG/ DIH+1 /1.75/1.72/1.63/1.6/1.55/1.49/1.49/1.49 :ft
.E DURW3 020212 GMT DH150000 /DUS /VB/ DIH+1 /13.84 :V
.E HOMN8 020212 GMT DH160000 /DUS /VB/ DIH+1 /14.11 :V
```

Figure 9-3: Example of SHEF .E output

9.3.2 SHEFIT Output Format

Figure 9-4 shows an example of the HEC SHEFIT output format.

```
CE459D7E20011203110000   0 0 0 0 0 0 HP RZZ  1055.530 Z -1.00    0 0          0
CE459D7E20011203100000   0 0 0 0 0 0 HP RZZ  1055.530 Z -1.00    0 0          0
CE459D7E20011203090000   0 0 0 0 0 0 HP RZZ  1055.530 Z -1.00    0 0          0
CE459D7E20011203080000   0 0 0 0 0 0 HP RZZ  1055.530 Z -1.00    0 0          0
CE459D7E20011203070000   0 0 0 0 0 0 HP RZZ  1055.530 Z -1.00    0 0          0
CE459D7E20011203060000   0 0 0 0 0 0 HP RZZ  1055.530 Z -1.00    0 0          0
CE459D7E20011203050000   0 0 0 0 0 0 HP RZZ  1055.530 Z -1.00    0 0          0
CE459D7E20011203040000   0 0 0 0 0 0 HP RZZ  1055.530 Z -1.00    0 0          0
CE459D7E20011203030000   0 0 0 0 0 0 HP RZZ  1055.530 Z -1.00    0 0          0
CE459D7E20011203020000   0 0 0 0 0 0 HP RZZ  1055.520 Z -1.00    0 0          0
CE459D7E20011203010000   0 0 0 0 0 0 HP RZZ  1055.520 Z -1.00    0 0          0
CE459D7E20011203000000   0 0 0 0 0 0 HP RZZ  1055.520 Z -1.00    0 0          0
CE459D7E20011203110000   0 0 0 0 0 0 PC RZZ      .000 Z -1.00    0 0          0
CE459D7E20011203100000   0 0 0 0 0 0 PC RZZ      .000 Z -1.00    0 0          0
CE459D7E20011203090000   0 0 0 0 0 0 PC RZZ      .000 Z -1.00    0 0          0
CE459D7E20011203080000   0 0 0 0 0 0 PC RZZ      .000 Z -1.00    0 0          0
CE459D7E20011203070000   0 0 0 0 0 0 PC RZZ      .000 Z -1.00    0 0          0
CE459D7E20011203060000   0 0 0 0 0 0 PC RZZ      .000 Z -1.00    0 0          0
```

Figure 9-4: Example of SHEFIT Output Format.

9.3.3 Human Readable Output Format

```
Message for Platform NWSHB5-HOMN8
                     |   elev   |    PC    | battery  |
                     |    HP    |    PC    |    VB    |
                     |    ft    |    in    |    V     |
12/03/2001 00:00:00 |  1055.53 | 0.0      |          |
12/03/2001 01:00:00 |  1055.53 | 0.0      |          |
12/03/2001 02:00:00 |  1055.53 | 0.0      |          |
12/03/2001 03:00:00 |  1055.53 | 0.0      |          |
12/03/2001 04:00:00 |  1055.53 | 0.0      |          |
12/03/2001 05:00:00 |  1055.53 | 0.0      |          |
12/03/2001 06:00:00 |  1055.53 | 0.0      |          |
12/03/2001 07:00:00 |  1055.53 | 0.0      |          |
12/03/2001 08:00:00 |  1055.53 | 0.0      |          |
12/03/2001 09:00:00 |  1055.52 | 0.0      |          |
12/03/2001 10:00:00 |  1055.52 | 0.0      |          |
12/03/2001 11:00:00 |  1055.52 | 0.0      |  13.876  |

Message for Platform NWSHB5-WTSM5
                     |   pool   |   tail   | battery  |
                     |    HP    |    HT    |    VB    |
                     |    ft    |    ft    |   VOLT   |
12/03/2001 00:00:00 |  900.0   |  935.5   |          |
12/03/2001 01:00:00 |  900.0   |  935.49  |          |
12/03/2001 02:00:00 |  900.0   |  935.5   |          |
12/03/2001 03:00:00 |  900.0   |  935.51  |          |
12/03/2001 04:00:00 |  900.0   |  935.54  |          |
12/03/2001 05:00:00 |  900.0   |  935.61  |          |
12/03/2001 06:00:00 |  900.0   |  935.65  |          |
12/03/2001 07:00:00 |  900.0   |  935.67  |          |
12/03/2001 08:00:00 |  900.0   |  935.67  |          |
12/03/2001 09:00:00 |  900.0   |  935.65  |          |
12/03/2001 10:00:00 |  900.0   |  935.64  |          |
12/03/2001 11:00:00 |  900.0   |  935.61  |  12.004  |
```

Figure 9-5: Example of Human Readable Output Format.

9.3.4 EMIT-ASCII Format

If the routing spec contains a string property called 'delimiter', this will be used to delimit between columns. The default is a single space.

The EMIT-ASCII formatter produces an output that is compatible with the old EMIT program when "ASCII" was selected as the output format. This format has 12 blank-delimited fields as follows:

- Hex DCP Address
- EPA Sensor Code (0 if none is assigned)
- Sensor Number
- Time Stamp in the format: YYDDD/HH:MM:SS
- Sample Value (formatted as specified by Presentation Group)
- 'I' if this is a self-timed message (meaning interval data); or 'R' if this is a random message.
- DCP Name (the preferred site name as specified by your properties file is used)
- Sensor Name
- SHEF Code (or 'XX' if none is specified)
- Recording interval for this sensor (in seconds)
- 'I'

Engineering Units

Following all sample data, a single line with 'ZZZZ' is printed. Figure 9-6 shows a single message in EMIT-ASCII format.

```
CE459D7E 0     1    01337/11:00:00  1055.53   I HOMN8      elev     HP 3600 I ft
CE459D7E 0     1    01337/10:00:00  1055.53   I HOMN8      elev     HP 3600 I ft
CE459D7E 0     1    01337/09:00:00  1055.53   I HOMN8      elev     HP 3600 I ft
CE459D7E 0     1    01337/08:00:00  1055.53   I HOMN8      elev     HP 3600 I ft
CE459D7E 0     1    01337/07:00:00  1055.53   I HOMN8      elev     HP 3600 I ft
CE459D7E 0     1    01337/06:00:00  1055.53   I HOMN8      elev     HP 3600 I ft
CE459D7E 0     1    01337/05:00:00  1055.53   I HOMN8      elev     HP 3600 I ft
CE459D7E 0     1    01337/04:00:00  1055.53   I HOMN8      elev     HP 3600 I ft
CE459D7E 0     1    01337/03:00:00  1055.53   I HOMN8      elev     HP 3600 I ft
CE459D7E 0     1    01337/02:00:00  1055.52   I HOMN8      elev     HP 3600 I ft
CE459D7E 0     1    01337/01:00:00  1055.52   I HOMN8      elev     HP 3600 I ft
CE459D7E 0     1    01337/00:00:00  1055.52   I HOMN8      elev     HP 3600 I ft
CE459D7E 00045 2    01337/11:00:00 0.0        I HOMN8      PC       PC 3600 I in
CE459D7E 00045 2    01337/10:00:00 0.0        I HOMN8      PC       PC 3600 I in
CE459D7E 00045 2    01337/09:00:00 0.0        I HOMN8      PC       PC 3600 I in
CE459D7E 00045 2    01337/08:00:00 0.0        I HOMN8      PC       PC 3600 I in
CE459D7E 00045 2    01337/07:00:00 0.0        I HOMN8      PC       PC 3600 I in
CE459D7E 00045 2    01337/06:00:00 0.0        I HOMN8      PC       PC 3600 I in
CE459D7E 00045 2    01337/05:00:00 0.0        I HOMN8      PC       PC 3600 I in
CE459D7E 00045 2    01337/04:00:00 0.0        I HOMN8      PC       PC 3600 I in
CE459D7E 00045 2    01337/03:00:00 0.0        I HOMN8      PC       PC 3600 I in
CE459D7E 00045 2    01337/02:00:00 0.0        I HOMN8      PC       PC 3600 I in
CE459D7E 00045 2    01337/01:00:00 0.0        I HOMN8      PC       PC 3600 I in
CE459D7E 00045 2    01337/00:00:00 0.0        I HOMN8      PC       PC 3600 I in
CE459D7E 70969 3    01337/11:00:00  13.876    I HOMN8      battery  VB 3600 I V
ZZZZ
```

Figure 9-6: Example of EMIT-ASCII format.

9.3.5 EMIT-Oracle Format

This format is similar to EMIT-ASCII but more compact. It was originally designed to input data into an Oracle database, hence the name. It is, however, a generally useful format in its own right, very easy to parse with a computer program.

The 'delimiter' property is supported in the same way as for EMIT-ASCII.

The EMIT-ORACLE formatter produces an output that is compatible with the old EMIT program when "ORACLE" was selected as the output format. This format has 7 blank-delimited fields as follows:

- Hex DCP Address
- SHEF Code (or 'XX' if none is specified)
- Sensor Number
- Time Stamp in the format: YYDDD/HH:MM:SS
- Sample Value (formatted as specified by Presentation Group)
- 'I' if this is a self-timed message (meaning interval data); or 'R' if this is a random message.

Engineering Units

Following all sample data, a single line with 'ZZZZ' is printed. Figure 9-7 shows a single message in EMIT-Oracle format.

```
CE459D7E HP 1    01337/11:00:00  1055.53   I ft
CE459D7E HP 1    01337/10:00:00  1055.53   I ft
CE459D7E HP 1    01337/09:00:00  1055.53   I ft
CE459D7E HP 1    01337/08:00:00  1055.53   I ft
CE459D7E HP 1    01337/07:00:00  1055.53   I ft
CE459D7E HP 1    01337/06:00:00  1055.53   I ft
CE459D7E HP 1    01337/05:00:00  1055.53   I ft
CE459D7E HP 1    01337/04:00:00  1055.53   I ft
CE459D7E HP 1    01337/03:00:00  1055.53   I ft
CE459D7E HP 1    01337/02:00:00  1055.52   I ft
CE459D7E HP 1    01337/01:00:00  1055.52   I ft
CE459D7E HP 1    01337/00:00:00  1055.52   I ft
CE459D7E PC 2    01337/11:00:00 0.0        I in
CE459D7E PC 2    01337/10:00:00 0.0        I in
CE459D7E PC 2    01337/09:00:00 0.0        I in
CE459D7E PC 2    01337/08:00:00 0.0        I in
CE459D7E PC 2    01337/07:00:00 0.0        I in
CE459D7E PC 2    01337/06:00:00 0.0        I in
CE459D7E PC 2    01337/05:00:00 0.0        I in
CE459D7E PC 2    01337/04:00:00 0.0        I in
CE459D7E PC 2    01337/03:00:00 0.0        I in
CE459D7E PC 2    01337/02:00:00 0.0        I in
CE459D7E PC 2    01337/01:00:00 0.0        I in
CE459D7E PC 2    01337/00:00:00 0.0        I in
CE459D7E VB 3    01337/11:00:00  13.876    I V
ZZZZ
```

Figure 9-7: Example of Emit-Oracle Output Format.

9.3.6 Dump Formatter

DumpFormatter is useful for testing and trouble-shooting. It dumps the raw message, performance measurements, and decoded data to an output interface. Figure 9-8 shows an example of this format.

```
================================
Start of message for platform NWSHB5-HOMN8
Time Stamp: 12/02/2001 16:08:11
Raw Data:
CE459D7E01336210811G44-
4NN031E9200077B1HAvq@@@Avq@@@Avq@@@Avq@@@Avq@@@Avq@@@Avq@@@Avq@@@Avq@@@Avq@@@Avp@@@Avp
@@@Avp@@@N

Performance Measurements:
DcpAddress=CE459D7E
Spacecraft=E
UplinkCarrier=92
Channel=31
SignalStrength=44
Length=77
ModulationIndex=N
Quality=N
Time=12/02/2001 21:08:11
FailureCode=G
FrequencyOffset=-4

Decoded Data:

Sensor 1: elev, EU=ft(feet), DataType=SHEF-PE:HP
Begin=12/02/2001 16:53:33, End=12/03/2001 06:00:00
Number of Samples=12
Sample[0]=12/03/2001 06:00:00: 1055.53  ' 1055.53'
Sample[1]=12/03/2001 05:00:00: 1055.53  ' 1055.53'
Sample[2]=12/03/2001 04:00:00: 1055.53  ' 1055.53'
Sample[3]=12/03/2001 03:00:00: 1055.53  ' 1055.53'
Sample[4]=12/03/2001 02:00:00: 1055.53  ' 1055.53'
Sample[5]=12/03/2001 01:00:00: 1055.53  ' 1055.53'
Sample[6]=12/03/2001 00:00:00: 1055.53  ' 1055.53'
Sample[7]=12/02/2001 23:00:00: 1055.53  ' 1055.53'
Sample[8]=12/02/2001 22:00:00: 1055.53  ' 1055.53'
Sample[9]=12/02/2001 21:00:00: 1055.52  ' 1055.52'
Sample[10]=12/02/2001 20:00:00: 1055.52  ' 1055.52'
Sample[11]=12/02/2001 19:00:00: 1055.52  ' 1055.52'
Sensor 2: PC, EU=in(inches), DataType=SHEF-PE:PC
Begin=12/02/2001 16:53:33, End=12/03/2001 06:00:00
Number of Samples=12
Sample[0]=12/03/2001 06:00:00: 0  '0.0     '
Sample[1]=12/03/2001 05:00:00: 0  '0.0     '
Sample[2]=12/03/2001 04:00:00: 0  '0.0     '
Sample[3]=12/03/2001 03:00:00: 0  '0.0     '
Sample[4]=12/03/2001 02:00:00: 0  '0.0     '
Sample[5]=12/03/2001 01:00:00: 0  '0.0     '
Sample[6]=12/03/2001 00:00:00: 0  '0.0     '
Sample[7]=12/02/2001 23:00:00: 0  '0.0     '
Sample[8]=12/02/2001 22:00:00: 0  '0.0     '
Sample[9]=12/02/2001 21:00:00: 0  '0.0     '
Sample[10]=12/02/2001 20:00:00: 0  '0.0     '
Sample[11]=12/02/2001 19:00:00: 0  '0.0     '
Sensor 3: battery, EU=V(volts), DataType=SHEF-PE:VB
Begin=12/02/2001 16:53:33, End=12/03/2001 06:00:00
Number of Samples=1
Sample[0]=12/03/2001 06:00:00: 13.876  ' 13.876
```

Figure 9-8: Example of Dump Output Format

9.3.7 USGS STDFMT Output Formatter

This formatter is used by USGS for ingesting data into the National Water Information System (NWIS). For documentation on this format see Appendix E in the NWIS User Guide, which can be found at:

`http://wa.water.usgs.gov/realtime/adaps/adaps.book.html`

Figure 9-9 shows an example of STDFMT output. Each DCP message is placed in a separate STDFMT envelope.

```
BE STDDCP
DB 1 1
SD  USGS       03323500  0N
SE    8   STAGE00065   11 73F010000
TM 20021030130000
UF   8 10.390 10.390 10.390 10.390 10.390 10.390 10.390 10.390
SE    3   H2O T00010   11 73F010000
TM 20021030130000
UF   8 10.600 10.600 10.600 10.700 11.100 11.400 11.600 11.800
SE    9   BATVT70969   11 63F010000
TM 20021030130000
UF   8 3.740 3.740 3.770 3.850 4.210 4.100 4.000 4.110
EE
BE STDDCP
DB 1 1
SD  USGS       03324500  0N
SE    2  STAGE200065   11 63F010000
TM 20021030130000
UF   8 4.540 4.540 4.540 4.540 4.540 4.540 4.540 4.540
SE    5  PREC 200045    6 83F010000
TM 20021030130000
UF   8 116.700 116.700 116.700 116.700 116.700 116.700 116.700 116.700
SE    3  H2O T100010   11 73F010000
TM 20021030130000
UF   8 13.000 13.000 13.000 13.000 13.000 13.000 13.000 13.000
SE    8  BATVTB70969   11 73F010000
TM 20021030130000
UF   8 15.310 15.210 14.910 14.720 14.830 14.710 14.710 14.830
EE
```

Figure 9-9: Example of USGS STDFMT Output.

9.3.8 Transmit Monitor Formatter

The Transmit Monitor format provides a log of transmission quality measurements in an easy-to-use row column format. The following columns are used by default:

- Message Time Stamp in the form MM/DD/YYYY-HH:MM:SS
- DCP Address (Transport Medium ID)
- Site Name
- Failure Code
- SignalStrength
- Message Length
- GOES Channel Number
- Frequency Offset
- Modulation Index

Battery Voltage

An example of the default format is shown in Figure 9-10.

```
10/30/2002-20:03:33 CE7718EE   03324500 G   50 209  23 -4  N  N 14.83
10/30/2002-20:16:50 CE77835E   03360000 G   50  97  23 -5  N  N 13.88
10/30/2002-20:29:25 CE777D08   03327500 G   50 161  23 -2  N  N 14.37
10/30/2002-21:03:11 CE14B3F8   03324000 G   50 145  17  0  H  N 13.70
10/30/2002-21:07:22 CE14C568   03275000 G   50 113  17 -1  N  N 13.74
10/30/2002-22:21:29 CE6D361C   03335500 G   49 113  41 -4  N  F 14.12
10/31/2002-00:03:33 CE7718EE   03324500 G   49 209  23 -3  N  N 14.72
10/31/2002-00:05:30 CE772D74   03375500 G   49 105  23 -3  N  N  13.3
10/31/2002-00:06:27 CE7730D0   03276000 G   49 145  23  2  N  N  14.8
10/31/2002-00:16:50 CE77835E   03360000 G   49  97  23 -5  N  N 14.23
10/31/2002-00:29:25 CE777D08   03327500 G   50 161  23 -2  N  N 13.99
10/31/2002-01:03:11 CE14B3F8   03324000 G   50 145  17  0  H  N 13.70
10/31/2002-02:21:29 CE6D361C   03335500 G   50 113  41 -4  N  N 14.11
10/31/2002-04:16:50 CE77835E   03360000 G   49  97  23 -5  N  N 13.88
10/31/2002-04:29:25 CE777D08   03327500 G   49 161  23 -2  N  N 13.79
10/31/2002-05:03:11 CE14B3F8   03324000 G   49 145  17  0  H  N 13.70
10/31/2002-05:05:41 CE14D61E   03357500 G   50 145  17 -9  N  N 14.70
10/31/2002-05:07:22 CE14C568   03275000 G   50 113  17 -1  N  N 13.57
10/31/2002-06:21:29 CE6D361C   03335500 G   50 113  41 -4  N  N 14.17
```

Figure 9-10: Example of Transmit Monitor Format.

You can control the contents of the transmit monitor format by adding properties to the routing specification:

- The string property “delimiter” has a default value of a single space character. This is used to separate columns in the output. To ingest this data into a SQL database, for example, you may wish to use a comma as a delimiter.

The Boolean property “justify” defaults to ‘true’. This causes each column to be either right or left justified within the column width. The example above shows justified columns.

The string property “columns” is a blank or comma-separated list of columns that you wish to see in the output. Table 9-1 shows the column names that can be included in this string. The default value for the string is:

```
"time id name FailureCode SignalStrength Length Channel FrequencyOffset
ModulationIndex Quality batt"
```

Column Name	*Description*
time	Message time stamp in the format MM/DD/YYYY-HH:MM:SS
id	Transport ID (i.e. DCP address for GOES messages)
name	Site name
FailureCode	1-character code for GOES messages: 'G' means good message, '?' means parity errors.
Length	Length of the raw message in bytes
Channel	GOES Channel number
FrequencyOffset	A sign plus a digit, taken from the DOMSAT message header, this indicates the frequency offset of the raw message, as reported by DAPS. The digit indicates the amount of the offset in units of 50Hz.
ModulationIndex	'N' for Normal, 'L' for Low, 'H' for High
Quality	'N' (normal) = Error rate betterh than 10^{-6}, 'F' (fair) = Error rate between 10^{-4} and 10^{-6} 'P' (poor) = Error rate worse than 10^{-4}
SignalStrength	in dB.
Spacecraft	'E' (East), or 'W' (West)
UplinkCarrier	Uplink Carrier Status (not implemented in DAPS-I)
batt	Battery voltage if available. The most recent sample contained in the message will be printed. This looks for a sensor with a name that starts with "batt". If none found it looks for any sensor with a datatype equivalent to VB.

Table 9-1: Column Names supported by Transmit Monitor Formatter.

The string property "colwidths" is used to control the width and justification of each column. It should be a blank or comma-separated list of numbers, one for each column. A positive number means right-justified. A negative number means left-justified. The default value of this property is:

```
19, 8, 10, 1, 5, 3, 2, 2, 2, 5
```

Example: Cause the formatter to print a comma-separated list of messages. For each message we only want the time, DCP Address, and battery voltage.

Add the following properties to the routing spec:

- delimiter = , (i.e. a single comma)
- justify = false
- columns = time id batt
- colwidths = 19, 8, 7

9.4 Consumers

Consumers receive the formatted data created by DECODES and send it somewhere. There are currently 4 types of consumers implemented within DECODES:

- PipeConsumer is used to send data from a DECODES routing spec into some other program in real time.
- FileConsumer sends all data from a routing spec into a single file. The file is closed when the routing spec is complete.
- DirectoryConsumer creates separate files for each message in a specified directory.

StringBufferConsumer is used internally by GUI programs that display decoded data interactively.

9.4.1 Pipe Consumer

The Consumer Argument should be one of:

- 'stdout' – send data to standard output.
- 'stderr' – send data to standard error.

command - an arbitrary command line. The command will be executed and the data will be piped to the command's standard input.

PipeConsumer will use the following routing-spec properties to control its actions:

Property Name	*Default*	*Description*
ConsumerBefore	none	An encoded string that is written to the file preceding each message. The string may contain UNIX-style escape sequences such as \n \r \t, and octal binary characters encoded as \002, etc.
ConsumerAfter	none	An encoded string that is written to the file after each message.

9.4.2 File Consumer

The Consumer Argument should be the file name template. The file will be opened when the routing spec starts. All data from the routing spec will be placed in the file. The file will be closed when finished.

The file name template may contain a variable of the form $DATE(*format*) where *format* describes how the date/time stamp is to be formatted in the file name. It can contain any format handled by the Java class java.text.SimpleDateFormat, although since it is used as a filename, it should not contain spaces or other illegal characters.

See http://java.sun.com/j2se/1.4.1/docs/api/ for complete docs on SimpleDateFormat. Click on "java.text" in the upper left frame. Then click on SimpleDateFormat in the lower left frame.

For example, if Consumer Arg is "data-$DATE(yyyyMMdd-HHmmss)", this might result in a filename data-20031213-120000.

This consumer should therefore only be used with routing specs that run for a finite period of time. That is, the 'until' time should be specified if reading from an LRGS.

FileConsumer will use the following routing-spec properties to control its actions:

Property Name	*Default*	*Description*
ConsumerBefore	none	An encoded string that is written to the file preceding each message. The string may contain UNIX-style escape sequences such as \n \r \t, and octal binary characters encoded as \002, etc.
ConsumerAfter	none	An encoded string that is written to the file after each message.

9.4.3 Directory Consumer

The Consumer Argument should be a directory name. The routing spec will create a separate file in this directory to hold the data generated for each message. The file name will be in the following format:

```
SiteName-YYYYMMDDHHmmSS
```

Hence when looking at a sorted directory listing you will see each platform's files together in time order.

The Site Name used will be the default site name type defined in your DECODES properties file.

DirectoryConsumer will use the following routing-spec properties to control its actions:

Property Name	*Default*	*Description*
ConsumerBefore	none	An encoded string that is written to the file preceding each message. The string may contain UNIX-style escape sequences such as \n \r \t, and octal binary characters encoded as \002, etc.
ConsumerAfter	none	An encoded string that is written to the file after each message.
TriggerCommand	none	NOT YET IMPLEMENTED. This is a command that DECODES will execute after each file is generated and placed in the directory. The command will be passed the complete file path-name as an argument.
filename	none	A file-name template to override the default described above. (see below)

Files will be constructed in a temporary location and then moved to the named directory. Therefore, you can write a program to scan the directory for new files and be assured that all files in the directory are complete.

If you supply a filename property, it will be used to construct the filename, overriding the default described above. The template may contain variables of the following form:

- $DATE(*format*) - See the description of this in section 9.4.2.
- $TRANSPORTID - will be replaced by the DCP address.
- $SITENAME - will be replaced by the site name.

9.5 Time Tagging Data Samples

The "DataOrder" property can be set to 'A' (for Ascending) or 'D' for Descending.

- Ascending means that the oldest samples are first in the message. Successive samples for the same sensor have an ascending time.

Descending means that the newest samples are first in the message. Successive samples for the same sensor have a descending time.

The DataOrder property can appear in several entities. This is how DECODES determines the order for a given sensor:

- If there is a "DataOrder" property in the Equipment Model associated with the Platform Config. Set this as the default for all sensors. In most cases, this is all that is necessary.
- If there is a "DataOrder" property in the Equipment Model associated with the transport medium, this overrides the previous value. An example for using this would be a random message that reports time in a different order than self-timed messages.
- If there is a "DataOrder" property in the EquipmentModel associated with the ConfigSensor, use it as the value for that particular sensor. Use this if sensors report data in different orders.
- If there is a DataOrder property in the PlatformSensor entity, use it as the value for that particular sensor.

10. Specific Scenarios

10.1 How To Create a New Platform Specification

Create a Site for the New Platform

First create a Site for this platform. Recall that in DECODES a "Site" refers to the location. The Platform resides at the Site.

1. Start the Database Editor by typing 'dbedit'.
2. Press the 'Sites' tab.
3. Make sure the site you want to create doesn't already exist. Press the column headers to sort by the various name types. If the site already exists, skip ahead to "Create a Configuration".
4. Press the 'New' button at the bottom of the Site List Panel. This creates a new site record and opens it. You now see a Site Edit Panel with the newly created site.
5. A site must have at least one valid name. The new panel shows a site with your default name-type (probably NWSHB5) and the name "NewName". Change these to the proper (unique) name for the new site.
6. On the right side of the Site Edit Panel you can enter other descriptive information about the site, such as Latitude, Longitude, Nearest City, etc. At a minimum you should enter the correct time zone for this site.
7. At the bottom of the Site Edit Panel, press 'Commit' and the 'Close'. You should now see your new site in the site list.

Create an Equipment Model Record for the New Platform

Equipment Model records hold information about each vendor's platform. Your platform's configuration will be associated with an Equipment Model.

8. Press the 'Equipment' tab at the top of the editor window. You now see the Equipment Model List Tab.
9. Each equipment model is given a unique abbreviated name. For example "SU8004D" is the name for a Sutron 8004 DCP. Does the equipment model for your new platform already exist? If so, make a note of it and skip ahead to "Create a Configuration".
10. Press "New" at the bottom of the Equipment Model List Panel. You are prompted to enter a new name. There should be no spaces in the name and it must be unique. A new Equipment Model record is created and opened. You now see the open Equipment Model Edit Panel.
11. In the edit panel type the descriptive information about the equipment. The Type value must be set to "DCP.

12. Press the “Add” button in the Properties area. Create a property called “DataOrder” (no spaces). The value should be either **A** or **D**. ‘A’ means ascending, meaning that in messages from this platform, the oldest samples are transmitted first.
13. Press “Commit” and “Close” when you are finished with the Equipment Model.

Create a Configuration for the New Platform

14. Next you need to create a configuration for the new platform. Press the ‘Configs’ tab at the top of the editor. You should now see the Configuration List Panel.

Recall that in DECODES, most of the information about how to decode and time-tag data is part of the configuration. A configuration can be shared by several platforms. For example if you installed 8 Sutron 8200 platforms with exactly the same sensors, and they all are programmed to generate identically-formatted messages; then you only need to create one configuration that all 8 platforms can share.

Do you already have a configuration for a platform identical to the new platform? If so, you can use it. Find its name in the list and make a note of it. Then skip ahead to “Create the Platform Record”.

15. Press the New button at the bottom of the Configuration List Panel. You are prompted to enter a unique name for the configuration.

Rules and Conventions for Configuration Naming

The only hard rules for configuration names are:

The name must be unique. No two configurations can have the same name.
The name cannot contain any spaces. We recommend limiting the name to alphanumeric and hyphens.

You should establish an agency-wide convention for naming configurations. You want to be able to exchange configurations with other districts (and other agencies) without fear of a name clash.

The USGS has a well established naming convention that other agencies are encouraged to follow:

EquipmentModelName-Organization-Sequence

…where:

EquipementModelName is the abbreviation of the Equipment Model record associated with this configuration. Example "SU8004D" for Sutron 8004 DCP.
Organization is an abbreviation that uniquely identifies your organization and district. For example "ACEMD" might mean the U.S. Army Corps of Engineers Maryland District.
Sequence is simply a sequence number.

The following are examples of existing configuration names:

SU8200D-ILEX-005
HA555D-ACEAL-017
HA570D-ACETN-015

Enter the Sensor and Formatting Information

After creating a new configuration, a Config Edit Panel will be opened. This screen contains a lot of important information.

16. Press the 'Select' button to the right of "Equipment Model". Select the equipment model from the pop-up list.
17. In the text area, type a brief description for this configuration. The description might contain the number of sensors, SHEF codes, etc.
18. Press the 'Add' button in the Sensors area of the screen. The "Edit Config Sensor Dialog" appears. Fill out the form. Enter a name and data-type for the sensor. Enter the correct sampling time and interval (this is important for correctly time-tagging samples from each sensor). When finished, press OK.
19. Repeat the previous step for each sensor.
20. Press the 'Add' button in the Decoding Scripts area. The "Edit Decoding Script Dialog" appears. A Decoding Script tells DECODES how to extract samples from the raw message.
21. Enter a script name in the upper right of the dialog. Commonly used names are "ST" for self-timed GOES messages and "RD" for random GOES messages.
22. Enter the format statements manually. Order is important. The script will start with the first statement. Use the "Up" and "Dn" buttons to move format statements, if necessary.
23. In the "Sensor Unit Conversions" area of the screen, enter the units and conversions for each sensor. Enter the units abbreviation (see standard EU abbreviations table below).
24. If no conversion is necessary, leave algorithm set to "None". This is appropriate if the DCP reports values that are already in engineering units. For a linear conversion, select Algorithm=linear. Then enter A and B coefficients for the equation:

    EU = A * RAW + B

25. Test your script. Load a raw DCP message into the "Sample Message" text area. The "Load" button lets you load from a file. Even more convenient: Open a LRGS Browser window and cut & paste. The sample area must start with the DCP address at the beginning of the DOMSAT header in the DCP message. Any spaces or other characters before the DCP address will cause decoding to fail.
26. Press the "Decode" button. This will apply the format statements and unit conversions to the raw data in the sample area. See the results in the Decoded Data area.
27. Tweak the format statements and unit conversions, then press "Decode" again. Repeat until decoding is correct. Then press OK.

28. If your platform sends both self-timed and random messages, you will need to create a separate Decoding Script for each. We recommend calling the Self-Timed Script "ST" and the Random Script "RD".

Create the Platform Record

You are finally ready to create the platform!

29. Press the "Platforms" tab at the top of the editor.
30. Press the "New" button at the bottom of the screen. This creates a new (empty) platform record and opens it in a Platform Edit Panel.
31. Press the "Choose" button to the right of "Site". Select the new site that you created above.
32. Type a brief description for this platform. This will show up in the list of platforms for your own easy reference.
33. Press the "Choose" button to the right of "Config". Select the configuration that you prepared above. Click OK when the pop-up tells you about the dangers of changing the configuration assignment. You should now see a list of sensors in the "Platform Sensor Information" area.
34. Press the "Add" button on the right of the Transport Media area. Fill out the dialog. A **Transport Medium** is the glue that associates an incoming message with your platform records and your decoding scripts.
35. Under Medium Type, select "goes-self-timed".
36. Under Medium Identifier, type the 8-hex-digit DCP address.
37. Under Decoding Script, select the name of the self timed script (e.g. "ST").
38. Enter the channel numbers, and if this is for a self-timed message, enter the time and interval values. Then press OK.
39. If this platform also transmits random, press Add again, select "goes-random" for Medium Type and "RD" for Decoding Script. Re-enter the DCP address.
40. At the bottom of the Platform Edit Panel press "Commit" and then "Close".

Add the new Platform to a Network List

41. Press the “Netlists” tab at the top of the editor.
42. Highlight the list you want to edit and press “Open”.
43. In the Network List Edit panel, press the “Add” button on the right.
44. Select your new site from the pop-up list and press OK.
45. Press “Commit” and “Close” at the bottom of the screen.

Testing the new Platform in a Routing Spec

46. Run a routing spec that uses the network list that you modified.
47. In the output data, you should see data from your new platform.

Engineering Unit List

DECODES is delivered with a fairly complete list of engineering units that are used in hydrometeorologic applications. You may define additional EUs by editing the XML file. Please email any proposed changes/additions to the decodes Email forum so that, if appropriate, they can be included in a subsequent release.

When you enter unit values for sensors, use the abbreviation. Case is ignored so ft, Ft, FT all refer to feet.

Sorted By Name

Name	Abbr	Family	Measures
acre feet	acre*ft	English	volume
acres	acre	English	area
atmospheres	atm	Metric	pressure
bars	bar	Metric	pressure
british thermal unit	btu	English	energy
Calories	cal	English	energy
centimeters	cM	Metric	length
centimeters per second	cM/s	Metric	speed
counts	count	univ	count
cubic centimeter	cc	Metric	volume
cubic feet	ft^3	English	volume
cubic feet per second	cfs	English	flow
cubic feet per second	ft^3/s	English	flow
cubic inches	in^3	English	volume
cubic meter	m^3	Metric	volume
cubic meters per second	m^3/s	Metric	flow
days	day	univ	time
degrees Celsius	degC	Metric	temperature
degrees Fahrenheit	degF	English	temperature
degrees Kelvin	degK	Metric	temperature
dyn	dyn	Metric	force
ergs	erg	English	energy
feet	ft	English	length
feet per second	ft/s	English	speed
fluid ounce	floz	English	volume

foot-pounds per second	ft*lbf/s	English	power
gallon	gal	English	volume
grams	G	Metric	mass
grams per liter	g/L	Metric	concentration
horsepower	hp	English	power
hours	hr	univ	time
inches	in	English	length
inches of mercury	inHg	English	pressure
inches per second	in/s	English	speed
joules	j	Metric	energy
Kilocalories	kcal	English	energy
kilograms	kG	Metric	mass
kilojoules	kj	Metric	energy
kiloliter	kL	Metric	volume
kilometers	kM	Metric	length
kilometers per hour	kM/hr	Metric	speed
kilopascals	kpa	Metric	pressure
kilowatts	kW	Metric	power
liter	L	Metric	volume
meters	M	Metric	length
meters per second	M/s	Metric	speed
Metric ton	mt	Metric	mass
micrograms	uG	Metric	mass
micrograms per liter	uG/L	Metric	concentration
microliter	uL	Metric	volume
micrometers	uM	Metric	length
MicroMHOs per centimeter	uMHOs/cm	metric	conductance
MicroMHOs per centimeter	uMHO	metric	conductance
MicroMHOs per centimeter	uMHOs	metric	conductance
miles	mi	English	length
miles per hour	mi/hr	English	speed
miles per hour	mph	English	speed
millibars	mbar	Metric	pressure
milligrams	mG	Metric	mass
milligrams per liter	mG/L	Metric	concentration
milliliter	mL	Metric	volume

millimeters	mM	Metric	length
millimeters of mercury	mmHg	Metric	pressure
millimeters per second	mM/s	Metric	speed
minutes	min	univ	time
nautical miles	nmi	English	length
nautical miles per hour	nmi/hr	English	speed
nautical miles per hour	knots	Metric	speed
newtons	N	Metric	force
ounces	oz	English	mass
parts per million	ppm	univ	ratio
parts per thousand	ppt	univ	ratio
pascals	pa	Metric	pressure
percent	pct	univ	ratio
percent	%	univ	ratio
pH	pH	univ	acidity
pint	pt	English	volume
pound-force	lbf	English	force
pounds	lb	English	mass
pounds per square inch	psi	English	pressure
quart	qt	English	volume
second	sec	univ	time
square centimeters	cM^2	Metric	area
square feet	ft^2	English	area
square inches	in^2	English	area
square kilometers	kM^2	Metric	area
square meters	M^2	Metric	area
square miles	mi^2	English	area
square millimeters	mM^2	Metric	area
square yards	yd^2	English	area
tons	ton	English	mass
volts	V	Metric	emf
watts	W	Metric	power
weeks	week	univ	time
yards	yd	English	length
yards per second	yd/s	English	speed

Sorted By Abbreviation

Name	Abbr	Family	Measures
percent	%	univ	ratio
acres	acre	English	area
acre feet	acre*ft	English	volume
atmospheres	atm	Metric	pressure
bars	bar	Metric	pressure
british thermal unit	btu	English	energy
Calories	cal	English	energy
cubic centimeter	cc	Metric	volume
cubic feet per second	cfs	English	flow
centimeters	cM	Metric	length
centimeters per second	cM/s	Metric	speed
square centimeters	cM^2	Metric	area
counts	count	univ	count
days	day	univ	time
degrees Celsius	degC	Metric	temperature
degrees Fahrenheit	degF	English	temperature
degrees Kelvin	degK	Metric	temperature
dyn	dyn	Metric	force
ergs	erg	English	energy
fluid ounce	floz	English	volume
feet	ft	English	length
foot-pounds per second	ft*lbf/s	English	power
feet per second	ft/s	English	speed
square feet	ft^2	English	area
cubic feet	ft^3	English	volume
cubic feet per second	ft^3/s	English	flow
grams	G	Metric	mass
grams per liter	g/L	Metric	concentration
gallon	gal	English	volume
horsepower	hp	English	power
hours	hr	univ	time
inches	in	English	length
inches per second	in/s	English	speed

square inches	in^2	English	area
cubic inches	in^3	English	volume
inches of mercury	inHg	English	pressure
joules	j	Metric	energy
Kilocalories	kcal	English	energy
kilograms	kG	Metric	mass
kilojoules	kj	Metric	energy
kiloliter	kL	Metric	volume
kilometers	kM	Metric	length
kilometers per hour	kM/hr	Metric	speed
square kilometers	kM^2	Metric	area
nautical miles per hour	knots	Metric	speed
kilopascals	kpa	Metric	pressure
kilowatts	kW	Metric	power
liter	L	Metric	volume
pounds	lb	English	mass
pound-force	lbf	English	force
meters	M	Metric	length
meters per second	M/s	Metric	speed
square meters	M^2	Metric	area
cubic meter	m^3	Metric	volume
cubic meters per second	m^3/s	Metric	flow
millibars	mbar	Metric	pressure
milligrams	mG	Metric	mass
milligrams per liter	mG/L	Metric	concentration
miles	mi	English	length
miles per hour	mi/hr	English	speed
square miles	mi^2	English	area
minutes	min	univ	time
milliliter	mL	Metric	volume
millimeters	mM	Metric	length
millimeters per second	mM/s	Metric	speed
square millimeters	mM^2	Metric	area
millimeters of mercury	mmHg	Metric	pressure
miles per hour	mph	English	speed

Metric ton	mt	Metric	mass
newtons	N	Metric	force
nautical miles	nmi	English	length
nautical miles per hour	nmi/hr	English	speed
ounces	oz	English	mass
pascals	pa	Metric	pressure
percent	pct	univ	ratio
pH	pH	univ	acidity
parts per million	ppm	univ	ratio
parts per thousand	ppt	univ	ratio
pounds per square inch	psi	English	pressure
pint	pt	English	volume
quart	qt	English	volume
second	sec	univ	time
tons	ton	English	mass
micrograms	uG	Metric	mass
micrograms per liter	uG/L	Metric	concentration
microliter	uL	Metric	volume
micrometers	uM	Metric	length
MicroMHOs per centimeter	uMHO	metric	conductance
MicroMHOs per centimeter	uMHOs	metric	conductance
MicroMHOs per centimeter	uMHOs/cm	metric	conductance
volts	V	Metric	emf
watts	W	Metric	power
weeks	week	univ	time
yards	yd	English	length
yards per second	yd/s	English	speed
square yards	yd^2	English	area

Sorted By Family, Name

Name	Abbr	Family	Measures
acre feet	acre*ft	English	volume
acres	acre	English	area
british thermal unit	btu	English	energy
Calories	cal	English	energy
cubic feet	ft^3	English	volume
cubic feet per second	cfs	English	flow
cubic feet per second	ft^3/s	English	flow
cubic inches	in^3	English	volume
degrees Fahrenheit	degF	English	temperature
ergs	erg	English	energy
feet	ft	English	length
feet per second	ft/s	English	speed
fluid ounce	floz	English	volume
foot-pounds per second	ft*lbf/s	English	power
gallon	gal	English	volume
horsepower	hp	English	power
inches	in	English	length
inches of mercury	inHg	English	pressure
inches per second	in/s	English	speed
Kilocalories	kcal	English	energy
miles	mi	English	length
miles per hour	mi/hr	English	speed
miles per hour	mph	English	speed
nautical miles	nmi	English	length
nautical miles per hour	nmi/hr	English	speed
ounces	oz	English	mass
pint	pt	English	volume
pound-force	lbf	English	force
pounds	lb	English	mass
pounds per square inch	psi	English	pressure
quart	qt	English	volume
square feet	ft^2	English	area
square inches	in^2	English	area

square miles	mi^2	English	area
square yards	yd^2	English	area
tons	ton	English	mass
yards	yd	English	length
yards per second	yd/s	English	speed
atmospheres	atm	Metric	pressure
bars	bar	Metric	pressure
centimeters	cM	Metric	length
centimeters per second	cM/s	Metric	speed
cubic centimeter	cc	Metric	volume
cubic meter	m^3	Metric	volume
cubic meters per second	m^3/s	Metric	flow
degrees Celsius	degC	Metric	temperature
degrees Kelvin	degK	Metric	temperature
dyn	dyn	Metric	force
grams	G	Metric	mass
grams per liter	g/L	Metric	concentration
joules	j	Metric	energy
kilograms	kG	Metric	mass
kilojoules	kj	Metric	energy
kiloliter	kL	Metric	volume
kilometers	kM	Metric	length
kilometers per hour	kM/hr	Metric	speed
kilopascals	kpa	Metric	pressure
kilowatts	kW	Metric	power
liter	L	Metric	volume
meters	M	Metric	length
meters per second	M/s	Metric	speed
Metric ton	mt	Metric	mass
micrograms	uG	Metric	mass
micrograms per liter	uG/L	Metric	concentration
microliter	uL	Metric	volume
micrometers	uM	Metric	length
MicroMHOs per centimeter	uMHO	metric	conductance
MicroMHOs per centimeter	uMHOs	metric	conductance

MicroMHOs per centimeter	uMHOs/cm	metric	conductance
millibars	mbar	Metric	pressure
milligrams	mG	Metric	mass
milligrams per liter	mG/L	Metric	concentration
milliliter	mL	Metric	volume
millimeters	mM	Metric	length
millimeters of mercury	mmHg	Metric	pressure
millimeters per second	mM/s	Metric	speed
nautical miles per hour	knots	Metric	speed
newtons	N	Metric	force
pascals	pa	Metric	pressure
square centimeters	cM^2	Metric	area
square kilometers	kM^2	Metric	area
square meters	M^2	Metric	area
square millimeters	mM^2	Metric	area
volts	V	Metric	emf
watts	W	Metric	power
counts	count	univ	count
days	day	univ	time
hours	hr	univ	time
minutes	min	univ	time
parts per million	ppm	univ	ratio
parts per thousand	ppt	univ	ratio
percent	%	univ	ratio
percent	pct	univ	ratio
pH	pH	univ	acidity
second	sec	univ	time
weeks	week	univ	time

Sorted By Measured Quantity, Name

Name	Abbr	Family	Measures
pH	pH	univ	acidity
acres	acre	English	area
square centimeters	cM^2	Metric	area
square feet	ft^2	English	area
square inches	in^2	English	area
square kilometers	kM^2	Metric	area
square meters	M^2	Metric	area
square miles	mi^2	English	area
square millimeters	mM^2	Metric	area
square yards	yd^2	English	area
grams per liter	g/L	Metric	concentration
micrograms per liter	uG/L	Metric	concentration
milligrams per liter	mG/L	Metric	concentration
MicroMHOs per centimeter	uMHO	metric	conductance
MicroMHOs per centimeter	uMHOs	metric	conductance
MicroMHOs per centimeter	uMHOs/cm	metric	conductance
counts	count	univ	count
volts	V	Metric	emf
british thermal unit	btu	English	energy
Calories	cal	English	energy
ergs	erg	English	energy
joules	j	Metric	energy
Kilocalories	kcal	English	energy
kilojoules	kj	Metric	energy
cubic feet per second	cfs	English	flow
cubic feet per second	ft^3/s	English	flow
cubic meters per second	m^3/s	Metric	flow
dyn	dyn	Metric	force
newtons	N	Metric	force
pound-force	lbf	English	force
centimeters	cM	Metric	length
feet	ft	English	length
inches	in	English	length

kilometers	kM	Metric	length
meters	M	Metric	length
micrometers	uM	Metric	length
miles	mi	English	length
millimeters	mM	Metric	length
nautical miles	nmi	English	length
yards	yd	English	length
grams	G	Metric	mass
kilograms	kG	Metric	mass
Metric ton	mt	Metric	mass
micrograms	uG	Metric	mass
milligrams	mG	Metric	mass
ounces	oz	English	mass
pounds	lb	English	mass
tons	ton	English	mass
foot-pounds per second	ft*lbf/s	English	power
horsepower	hp	English	power
kilowatts	kW	Metric	power
watts	W	Metric	power
atmospheres	atm	Metric	pressure
bars	bar	Metric	pressure
inches of mercury	inHg	English	pressure
kilopascals	kpa	Metric	pressure
millibars	mbar	Metric	pressure
millimeters of mercury	mmHg	Metric	pressure
pascals	pa	Metric	pressure
pounds per square inch	psi	English	pressure
parts per million	ppm	univ	ratio
parts per thousand	ppt	univ	ratio
percent	%	univ	ratio
percent	pct	univ	ratio
centimeters per second	cM/s	Metric	speed
feet per second	ft/s	English	speed
inches per second	in/s	English	speed
kilometers per hour	kM/hr	Metric	speed

meters per second	M/s	Metric	speed
miles per hour	mi/hr	English	speed
miles per hour	mph	English	speed
millimeters per second	mM/s	Metric	speed
nautical miles per hour	nmi/hr	English	speed
nautical miles per hour	knots	Metric	speed
yards per second	yd/s	English	speed
degrees Celsius	degC	Metric	temperature
degrees Fahrenheit	degF	English	temperature
degrees Kelvin	degK	Metric	temperature
days	day	univ	time
hours	hr	univ	time
minutes	min	univ	time
second	sec	univ	time
weeks	week	univ	time
acre feet	acre*ft	English	volume
cubic centimeter	cc	Metric	volume
cubic feet	ft^3	English	volume
cubic inches	in^3	English	volume
cubic meter	m^3	Metric	volume
fluid ounce	floz	English	volume
gallon	gal	English	volume
kiloliter	kL	Metric	volume
liter	L	Metric	volume
microliter	uL	Metric	volume
milliliter	mL	Metric	volume
pint	pt	English	volume
quart	qt	English	volume